COOPERATIVE INSTITUTE FOR METEOROLOGICAL SATELLITE STUDIES
Space Science and Engineering Center
University of Wisconsin—Madison
1225 West Dayton Street
Madison, Wisconsin 53706
Telex: 265 452 UOFWISC MDS

UW-Madison.
SSEC Publication No.99.03.A1.

CIMSS

March 9, 1999

Dr. Gary Toller
SAIC/GSC
4600 Powder Mill Road
Suite 400
Beltsville, MD. 20705
Tel. 301 902-6032
FAX 301 931-3797
toller@gsc.saic.com

Dear Dr. Toller;

Enclosed is our final year interim report for the first phase of our cloud masking project for GLI. Included with the report are the ATBD and the cloud masking code delivered to EORC, which was integrated into the AVS/EXPRESS system the beginning of this year.

Sincerely

Steven A. Ackerman

Final Report

Pre-Launch Standard product algorithm development for the Global Imager aboard the Advanced Earth Observing Satellite II.

For the Period 01-08-97 thru 12-31-98
Agreement Identification Number G-0056

Dr. Steven A. Ackerman

Cooperative Institute for Meteorological Satellite Studies
Space Science and Engineering Center (SSEC)
at the University of Wisconsin-Madison
1225 West Dayton Street
Madison, Wisconsin 53706
608/262-0544

Table of Contents

INTRODUCTION

Final report for the Pre-launch Standard Product Cloud Mask Algorithm Development for GLI aboard ADEOS II. This report covers research activities of the GLI cloud masking algorithm (ATSK1, ATSK2) development and the cloud phase algorithm (ATSK4).

DESCRIPTION OF ACTIVITIES

ATSK1/ATSK2

Dr. Ackerman attended the 1st and 2nd GLI Workshops where the cloud masking algorithm was discussed with GLI scientists. During the second of these GLI Workshops, the 16 bit, 1km cloud mask was defined as a new Standard Product. The 16 bit structure is currently:

16 BIT GLI CLOUD MASK FILE SPECIFICATION		
BIT FIELD	**DESCRIPTION KEY**	**RESULT**
0-1	Unobstructed FOV Quality Flag	00 = cloudy 01 = probably clear 10 = confident clear 11 = high confidence clear
	PROCESSING PATH FLAGS	
2	Day / Night Flag	0 = Night / 1 = Day
3	Sun glint Flag	0 = Yes / 1 = No
4	Snow / Ice Background Flag	0 = Yes/ 1 = No
5-6	Land / Water Flag	00 = Water 01 = Coastal 10 = Desert 11 = Land
	ADDITIONAL FLAGS	
7	Non-cloud obstruction Flag (heavy aerosol)	0 = Yes / 1 = No
8	Thin Cirrus Detected (solar)	0 = Yes / 1 = No
9	Shadow Detected	0 = Yes / 1 = No
	1-km CLOUD FLAGS	
10	Result from Group I tests	0 = Yes / 1 = No
11	Result from Group II tests	0 = Yes / 1 = No
12	Result from Group III tests	0 = Yes / 1 = No
13	Result from Group IV tests	0 = Yes / 1 = No
14	Result from Group V tests	0 = Yes / 1 = No
15	Spare	

Dr. Ackerman and Mr. R. Frey visited EORC February 22-25, 1998 for the peer review of ATSK1 and ATSK4 and integration of ATSK1 to AVS. This meeting was very productive, it identified issues regarding the 1.6 μm channel and and the successful integration of ATSK1 into AVS.

The ATSK1 algorithm is discussed in the ATBD which is attached as Appendix A. The most recent version of the code is attached as Appendix B. This report presents an overview of the cloud masking approach and presents some validation approaches not given in the ATBD.

The GLI cloud mask will indicate whether a given view of the earth surface is unobstructed by clouds or optically thick aerosol, and whether that clear scene is contaminated by a shadow. The cloud mask will be generated at 1 km resolution. Input to the cloud mask algorithm is assumed to be calibrated and navigated level 1B radiance data. Additionally, the GLI data are assumed to be meeting specification.

The cloud mask will be determined for good data only. Incomplete or bad radiometric data will create holes in the cloud mask.

Several points need to be made regarding the approach to the GLI cloud mask:

(1) This is version 1.0 (version 2.0 is under development).
(2) The cloud mask algorithm assumes that calibrated, quality controlled data are the input and a cloud mask is the output.
(3) While the current version relies heavily on the MODIS cloud mask there are notable differences between the two approaches: the output of the GLI cloud mask is 16 bits (the MODIS is 48 bits) and is run on 1 km pixels only.

In the cloud masking with GLI, there are operational constraints to consider. These constraints are driven by the need to process GLI data in a timely fashion.

◊ CPU Constraint: Many algorithms must first determine if the pixel observation is cloudy or clear. A single cloud mask for GLI exists to avoid duplication in the many GLI algorithms. Thus, the cloud mask algorithm lies at the top of the data processing chain and must be versatile enough to satisfy the needs of many applications. The clear-sky determination algorithm must run in real time, limiting the use of CPU-intensive algorithms.

◊ Output File Size Constraint: Storage requirements are also a concern. The current cloud mask is more than a yes/no decision. The 16 bits of the mask include an indication of the likelihood that the pixel is contaminated with cloud. It also includes ancillary information regarding the processing path and the results from individual tests and other useful information (e.g., land/sea tag). In processing applications, one need not process all the bits of the mask. An algorithm can make use of only the first 8 bits of the mask if that is appropriate. The current mask requires approximately 1.5 gigabytes of storage per day.

◊ Comprehension: Because there are many users of the cloud mask, it is important that the mask not only provide enough information to be widely used but is also easily understood. To intelligently interpret the output from this algorithm, it is important to have the algorithm simple in concept but effective in its application.

The algorithm operates under different domains according to surface type and solar illumination: daytime land surface, daytime water, nighttime land, nighttime water, daytime desert, nighttime desert, and daytime and nighttime snow regions. Once a pixel has been assigned to a particular domain, the spectral data corresponding to that location are subjected to a series of threshold tests designed to detect the presence of clouds in the instrument FOV. These tests are the heart of the cloud mask algorithm. There are several types of tests, different cloud conditions relying on detection be different tests. For example, low cloud or thin cirrus cloud will be detected in one test, while optically thick cloud will be detected in another. Accordingly, tests which indicate similar cloud conditions are grouped together to obtain several intermediate results that are then combined to form a final cloud mask value. The tests are grouped so that independence between them is maximized. All tests which detect thin cirrus might make up a group, for example, while those which find high, cold clouds would form another, and low-level cloud detection tests could make a third set. Results from these groups are included in the cloud mask output. Of course, few if any spectral tests are completely independent of all the others.

The result of each spectral threshold test is expressed as a "confidence" which indicates the strength of the observed radiance signature compared to that which is expected for the cloud condition in question. For example, one fundamental test performed for water surfaces is the "cold cloud test." Over open waters any scene with an observed 11 μm brightness temperature colder than ~270°K must be at least partially cloudy. Therefore, the threshold value for this test is set at that value. The actual cutoff temperature for any given FOV varies slightly, however, because of differing amounts of atmospheric water vapor attenuation due to changes in actual water vapor content or instrument viewing angle. Consequently, a "confidence window" is constructed with boundaries at 267° and 273°K. Since the ultimate goal of the algorithm is to specify a confidence of clear-sky, an observed brightness temperature of < 267°K is defined to have 0 confidence while a measurement of > 273°K has a confidence 1.0. Note that this confidence does not yet refer to clear-sky conditions, only to the particular condition tested for. In this case it indicates only the extent of confidence that the FOV did *not* contain significant amounts of opaque,

cold clouds. Observations between 267° and 273°K result in confidences ranging linearly between 0 and 100%.

When all tests within a group have been performed, the minimum resulting confidence from among them is taken to be representative of that group. Then the other groups of tests are performed with group confidences determined in the same way. These confidences indicate absence of particular cloud types. A final step is to combine the group confidences, assumed to be independent, by multiplying them together and taking the Nth root, where N represents the number of groups. Only at this point does the confidence value reflect the surety of clear-sky conditions.

Using this algorithm, most observations have either high confidences (> 0.95) or very low confidences (< 0.05) of unobstructed views of the surface. There are always those difficult scenes, however, which result in intermediate values of the cloud mask. These tend to be found at cloud boundaries, or where low clouds are found over water surfaces at night, or over certain land surfaces such as desert or other sparsely-vegetated regions. In these cases (final confidence > 0.05 and < 0.95), spatial and/or temporal continuity tests are conducted. Spatial continuity testing is implemented for water surfaces. 11 μm brightness temperature differences between the pixel of interest and the surrounding eight are checked for consistency. If all the differences are less than 0.5 K, the confidence is adjusted upward by one "level" (e.g. > 0.66 to > 0.95). A "clear-sky data map" for testing temporal continuity is being developmed to better discriminate clear-sky ocean scenes.

Initial approaches to validating the cloud mask algorithm include visualization of the results in combination with spectral imagery, comparison with MODIS approach, comparison with surface cloud observations, and comparison with lidar retrievals. The beginning comparison studies are promising as they indicate the utility of the approach.

The GLI bands used in the version 1.0 of the cloud mask algorithm are:

Band	Wavelength (μm)	Used in Cloud Mask	
8	0.55	Y	thick clouds
13	0.68	Y	Clouds
19	0.88	Y	low clouds
26	1.24	Y	snow, clouds
27	1.38	Y	thin cirrus
28	1.64		
30	3.7	Y	Window
31	6.7	Y	high moisture
34	8.6	Y	mid moisture
35	10.8	Y	Window
36	12.0	Y	low moisture

Image visualization is an important component of validating any cloud detection algorithm. Additional validation includes comparison with active and passive measurements of clouds. In anticipation of future MODIS cloud mask verification efforts, a prototype methodology has been developed using the AVHRR cloud mask product. Three complete AVHRR GAC orbits are processed daily, using the previous day's level-1b data as input. Daytime coverage includes the Amazon Basin, Eastern North America, the Saudi Arabian Peninsula, and Central Europe. Hourly surface observations from ten manned weather stations in North America (ranging in latitude from 30N to 70N), closest in time to the satellite measurement, are collected and compared to the cloud mask output. Figure 1 is a 3-D histogram that compares the surface observation with the closest AVHRR pixel for August and September in 1997. Weather station cloud reports are plotted on the x-axis and the satellite derived clear sky confidence level on the y-axis. The z-axis is the frequency of occurrence in each category. Surface observations of cloudiness are reported as "clear", "few", "scattered", "broken" and "overcast". Cloud coverage for these categories is 0/8, 1/8-2/8, 3/8-4/8, 5/8-7/8, and 8/8, respectively. The cloud mask is generally doing a satisfactory job, particularly for clear skies. This is a result of the conservative nature of the cloud mask,

where only very high confidence pixels are designated as clear. As one sees more clouds from the ground, the cloud mask confidence for seeing clear sky diminishes, as expected.

To quantify the MAS cloud mask algorithm performance, comparisons are made with observations from the Cloud Lidar System (CLS) [*Spinhirne and Hart* 1990]. During the Subsonic Aircraft Contrail and Cloud Effects Special Study (SUCCESS), the MAS flew along with the lidar. The CLS algorithm detects a maximum of five cloud top and cloud bottom altitudes based upon the backscatter signal. Each collocation consists of the CLS cloud product and approximately 250 to 300 MAS pixels. The percentage of pixels labeled confident clear, probably clear, uncertain, and low confident clear are determined for each collocated scene. The CLS observations are divided into three categories:

Clear—no cloud was detected by the lidar;

Thin cloud—cloud boundary was detected, but a surface return signal was also received;

Thick cloud—cloud boundary was detected with no surface return signal.

Histograms of the percentage of pixels in a given confidence interval are plotted for each CLS cloud type category three days during SUCCESS (see Figure 2). Nearly all of the CLS labeled clear scenes are identified as high confident clear by the MAS cloud mask algorithm. Essentially all of the CLS labeled thick cloud scenes are labeled as cloudy by the MAS cloud mask. A majority of the thin cloud cases are labeled as either confident clear or cloudy by the MAS cloud mask algorithm. Differences between those scenes labeled clear and those classified cloudy are related to the cloud thickness. A more detailed analysis is required for verification of these thin clouds. Such a study is in progress. Also encouraging from this comparison is that few of the scenes are labeled as uncertain. Visualization of the cloud mask indicates that many of these scenes occur near cloud edges.

We have also compared results for the GLI cloud mask with the MODIS cloud mask using MAS observations. For snow free scenes there is no statistical difference between the GLI and MODIS results. Over snow scenes, as much as 25% of the pixels are assigned different level of confidences by the two approach. Most of the differences are one confidence level and result because of the use of the 1.6 μm channel in the MODIS approach. Figure 3 presents an image comparison of the MODIS and GLI cloud masking over a snow scene, this is the worst comparison observed to date. We are currently investigating how to best use the sub-sampled (every 4 km) 1.6 μm observations of the GLI in ATSK1.

ATSK4

We have begun work on adapted the MODIS infrared cloud phase algorithm to a GLI algorithm. Cloud phase will be obtained by MODIS 8, 11 and 12 μm brightness temperature difference at 5×5 pixel resolution (Level 2), and temporally averaged at .5 degree resolution (Level 3). The essence of the tri-spectral method consists of interpreting a scatter diagram of 8 minus 11 μm versus 11 minus 12 μm brightness temperature differences. For GLI we want to combine the IR approach with the visible/near-infrared approach using a fuzzy logic approach) to result in a single MODIS cloud phase product. This is a focus of our work over the next 6 months.

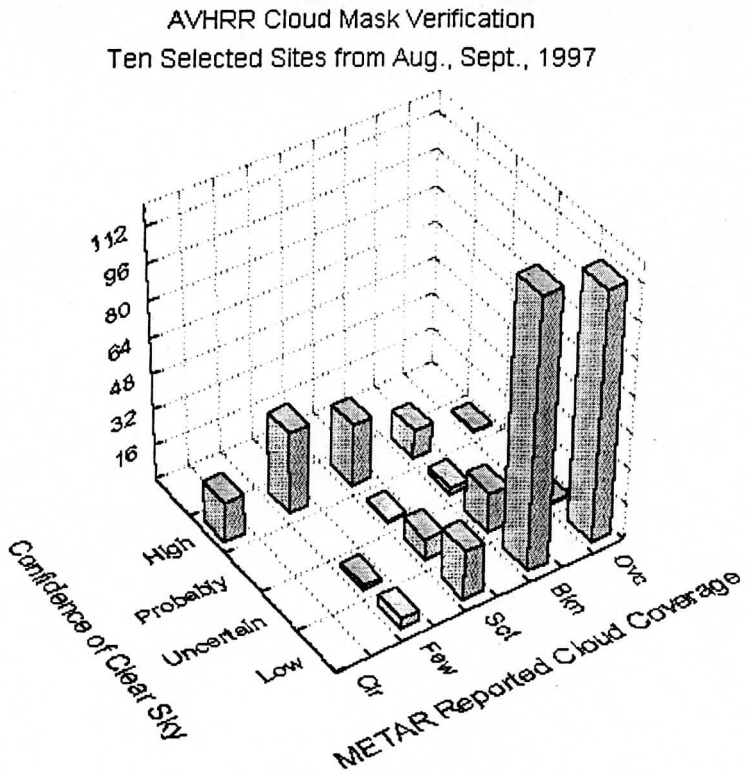

Figure 1. Comparison of surface cloud reports and AVHRR cloud mask results. Note the good agreement in clear as well as broken and overcast skies.

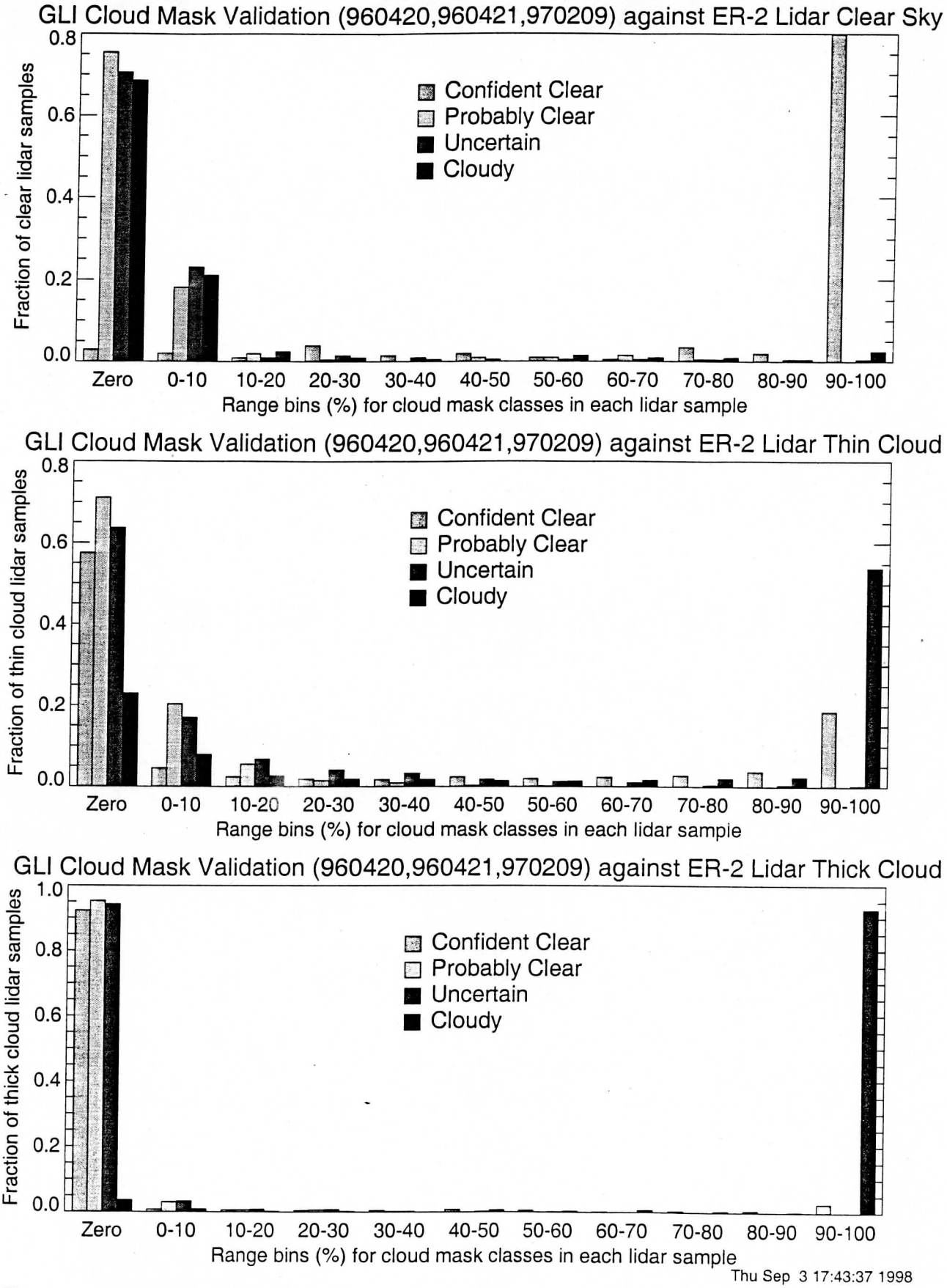

Figure 2. Comparisons between the cloud detection results using MAS and cloud lidar system for five days of the SUCCESS experiment. Cloud mask confidence is compared with three lidar categories: no cloud (top panel), thin cloud (middle panel), and thick cloud (bottom panel).

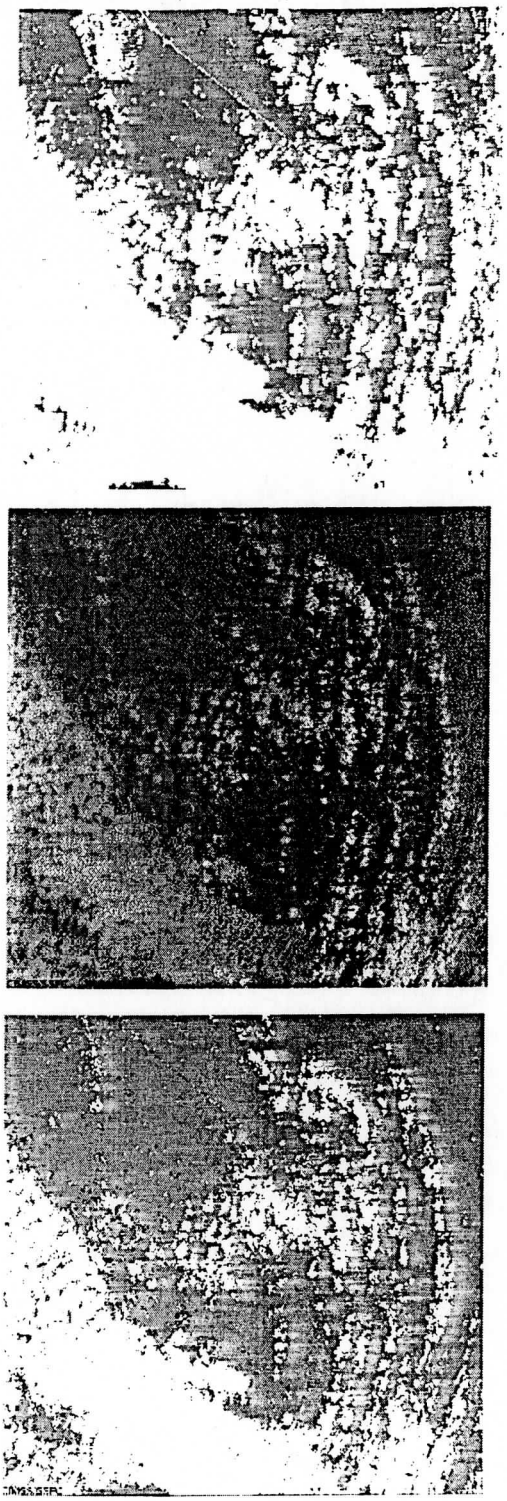

Figure 3. Comparison between the GLI cloud detection results and the MODIS results using MAS observations. This is the worst comparison, snow free comparison have no statistical differences between the two approaches.

ISSUES AND PROBLEMS

ATSK1/ATSK2

We have discussed the issue of using the 250m 1.64 μm channel in cloud masking with Dr. Takashi Nakajima and the EORC scientists. We conducted experiments with and without this channel in our cloud masking algorithm, and have removed it from the tests. The channel is particularly useful for separating clouds from clear when the surface is snow covered. Tradeoffs of data storage requirements and cloud masking were discussed with T. Nakajima. We initially agreed to forgo the use of the 1.64 μm channel. The 3.7 μm channel and ancillary data will be used to assist in detecting clouds over snow. However, more recent discussions indicate we should develop and approach that uses the sub-sampled 1.64 μm data. This will be one of the focuses of our upcoming GLI support.

ATSK4

Our focus over the next 6 months is the development and testing of this algorithm.

ALGORITHM STATUS

ATSK1/ATSK2

The first version of the cloud mask (ATSK1,ATSK2) has been developed and delivered to the EORC. The current status of ATSK1 is listed below:

Algorithm Code Name	Submit to EORC	Initial Check	Analysis I/O	Separation	AVS Conversion
ATSK1	97.11 Accepted	98.6.14 Finished	98.6.28 Finished	98.7.12 Finished	98.8.23 Finished

Version 1 of the GLI cloud mask was delivered on November 26 1997 to NASDA. We developed a version of the code that operates on MAS data, a NASA instrument that flies on the ER-2 and has many similar channels as the GLI. This code is available to GLI scientists. We worked with Dr. Junichi Inoue and his group in testing this code. Static analyzing the source programs was error free, compiling and linking was also completed without errors. The output cloud mask data calculated in EORC were coincident with the one delivered for both daytime and nighttime cases.

Version 1.0 of the Algorithm Theoretical Basis Document has been written which describes the GLI cloud mask and is attached. The ATBD is currently being up-dated.

ATSK1/ATSK2

The ATBD for the cloud phase (ATSK4) is currently under development. Version 1.0 of the ATBD and the code will be available in the summer 1999.

ACTIVITIES PLANNED FOR THE NEXT GLI RESEARCH PERIOD:

- continue validation of the GLI cloud mask,
- comparison of GLI mask with MODIS cloud mask using MAS and MODIS observations.
- determine threshold optical depth detected by the GLI mask.
- define the conditions for which the cloud mask algorithm will not be executed.
- preparation of the next version of the GLI cloud mask.
- assessment of using a clear sky radiance composite map,
- determine how to best use the GLI 1.64 μm channel.
- attend appropriate GLI workshops.

- develop and test ATSK4 (cloud phase algorithm).

SUMMARY

The pre-launch cloud/clear detection algorithm (version 1.0) has been developed and delivered to EORC. Further revisions and validation of the algorithm will be conducted during the next phase of the project. We will be also developing and implementing the pre-launch cloud phase algorithm (ATSK4).

Steve Ackerman
SSEC/CIMSS/AOS
1225 W. Dayton Street
University of Wisconsin-Madison
Madison, WI 53706
stevea@ssec.wisc.edu
Phone: 608-263-3647
Fax: 608-262-5974

APPENDIX A CLOUD FLAG PRODUCT

Product Name:	Product Code:
Cloud flags dataset(pixel by pixel)	**CLFLG_p**

DEFINITION

The GLI cloud mask will indicate whether a given view of the earth surface is unobstructed by clouds or optically thick aerosol, and whether that clear scene is contaminated by a shadow. The cloud mask will be generated at 1 km resolution. The cloud flag is more than a simple yes/no decision (though bit 1 alone could be used to represent a single bit cloud mask). The cloud mask includes 4 levels of 'confidence' with regard to whether a pixel is thought to be clear (bits 1 and 2) as well as the results from different spectral tests.

DESCRIPTION

(1) **Input**:

□Level-1B

(2) **Output**:

□Scene

(3) **Algorithm**:

□Used algorithms: ATSK1,2

(4) **Sample image:**

Nothing

SPECIFICATIONS

(1) **Geophysical Units**: none

(2) **Spatial resolution**: 1km

(3) **Temporal resolution**: every path, all scene

(4) **Location accuracy**: same as Level-1B data

(5) **Quality parameters**:
Refer to bit configuration in the format *(For reference I have attached it to this document)*

(6) **Other specific parameters**:
Number of total pixel
Ratio of 99% prob. clear pixel
Ratio of 95% prob. clear pixel
Ratio of 66% prob. clear pixel

(7) **Product size/Data volume**:
1km 1scene: 1276*1680*2byte/pixel = 4.1MB

(8) **Area coverage**: Scene(1600km*1680km)

(9) **Medium** *(TBD)*

(10) **Format** (*TBD*): Refer to the document **'GLI File Specifications'**, p.xx

Reference documents:

Ackerman, S. A., K. I. Strabala, W. P. Menzel, R. A. Frey, C. C. Moeller and L. E. Gumley, 1998: Discriminating Clear-sky from Clouds with MODIS *J. Geo. Res.*, **103** , D24, p. 32,141

King, M. D., S.-C. Tsay, S. A. Ackerman and N. F. Larsen, 1998: Discriminating Heavy Aerosol, Clouds, and Fires During SCAR-B: Application of Airborne Multispectral MAS Data,. Accepted for publication in to *J. Geo. Res.*

Riggs, G. A., D. K. Hall, S. A. Ackerman, 1998: Sea Ice Detection with the Moderate Resolution Imaging Spectroradiometer Airborne Simulator (MAS) Accepted for publication in Remote Sensing Environ.

(11)

APPENDIX B ATSK1/ATSK2 ATBD

DISCRIMINATING CLEAR-SKY FROM CLOUD WITH GLI

Steve Ackerman

(In collaboration with the MODIS Cloud Mask Team)

Cooperative Institute for Meteorological Satellite Studies, University of Wisconsin - Madison

Version 1.1

February 1999

TABLE OF CONTENTS

1.0 Introduction

Clouds are generally characterized by higher reflectance and lower temperature than the underlying earth surface. As such, simple visible and infrared window threshold approaches offer considerable skill in cloud detection. However, there are many surface conditions when this characterization of clouds is inappropriate, most notably over snow and ice. Additionally, some cloud types such as thin cirrus, low stratus at night, and small cumulus are difficult to detect because of insufficient contrast with the surface radiance. Cloud edges cause further difficulty since the instrument field of view will not always be completely cloudy or clear.

The 36 channel Global Imager (GLI) offers the opportunity for multispectral approaches to cloud detection so that many of these concerns can be mitigated. This document describes the approach and algorithms for detecting clouds using GLI observations. The algorithms have been drafted in close collaboration with members of the MODIS Science Team. A cloud mask algorithm that incorporates MAS observations to simulate the GLI cloud mask is available to the science team to the GLI masking approach. Section 2 gives an overview of the masking approach. The individual spectral tests are discussed in Section 3, and Section 4 discusses some practical applications.

2.0 Overview

2.1 *Objective*

The GLI cloud mask will indicate whether a given view of the earth surface is unobstructed by clouds or optically thick aerosol, and whether that clear scene is contaminated by a shadow. The cloud mask will be generated at 1 km resolution. Input to the cloud mask algorithm is assumed to be calibrated and navigated level 1B radiance data. Additionally, the GLI data are assumed to be meeting specification. The cloud mask will be determined for good data only. Incomplete or bad radiometric data will create holes in the cloud mask.

Several points need to be made regarding the approach to the GLI cloud mask presented in this document.

(1) This is version 1.0.

(2) The cloud mask algorithm assumes that calibrated, quality controlled data are the input and a cloud mask is the output.

(3) While the current version relies heavily on the MODIS cloud mask there are notable differences between the two approaches: the output of the GLI cloud mask is 16 bits (the MODIS is 48 bits) and is run on 1 km pixels only.

The snow bit in the cloud mask output indicates a processing path in the current algorithm and should not be considered the final indicator of this product. This is the first step in distinguishing cloud from snow. In certain heavy aerosol loading situations (e.g., dust storms, volcanic eruptions and forest fires) particular tests may flag the aerosol laden atmosphere as cloudy. An aerosol bit has initially been included in the mask to indicate fields of view that are potentially contaminated with optically thick aerosol.

In the cloud masking with GLI, there are operational constraints to consider. These constraints are driven by the need to process GLI data in a timely fashion.

◊ CPU Constraint: Many algorithms must first determine if the pixel observation is cloudy or clear. A single cloud mask for GLI exists to avoid duplication in the many GLI algorithms. Thus, the cloud mask algorithm lies at the top of the data processing chain and must be versatile enough to satisfy the needs of many applications. The clear-sky determination algorithm must run in real time, limiting the use of CPU-intensive algorithms.

- ◊ Output File Size Constraint: Storage requirements are also a concern. The current cloud mask is more than a yes/no decision. The 16 bits of the mask include an indication of the likelihood that the pixel is contaminated with cloud. It also includes ancillary information regarding the processing path and the results from individual tests and other useful information (e.g., land/sea tag). In processing applications, one need not process all the bits of the mask. An algorithm can make use of only the first 8 bits of the mask if that is appropriate. The current mask requires approximately 1.5 gigabytes of storage per day.
- ◊ Comprehension: Because there are many users of the cloud mask, it is important that the mask not only provide enough information to be widely used but is also easily understood. To intelligently interpret the output from this algorithm, it is important to have the algorithm simple in concept but effective in its application.

2.2 *Background*

Development of the GLI cloud mask algorithm benefits from previous work to characterize global cloud cover using satellite observations. It builds on the current MODIS cloud masking algorithm. Other approaches that have influenced the development of the MODIS/GLI mask are briefly discused in this section. The International Satellite Cloud Climatology Project (ISCCP) has developed cloud detection schemes using visible and infrared window radiances. The AVHRR (Advanced Very High Resolution Radiometer) Processing scheme Over cLoud Land and Ocean (APOLLO) cloud detection algorithm uses the five visible and infrared channels of the AVHRR. The NOAA Cloud Advanced Very High Resolution Radiometer (CLAVR) also uses a series of spectral and spatial variability tests to detect a cloud. CO_2 slicing characterizes global high cloud cover, including thin cirrus, using infrared radiances in the carbon dioxide sensitive portion of the spectrum. Additionally, spatial coherence of infrared radiances in cloudy and clear skies has been used successfully in regional cloud studies. The following paragraphs briefly summarize some of these prior approaches to cloud detection.

The International Satellite Cloud Climatology Project (ISCCP) cloud masking algorithm is described by Rossow (1989, 1993), Rossow et al. (1989), Seze and Rossow (1991) and Rossow and Gardner (1993). Only two channels are used in cloud detection, the narrowband visible (0.6 μm) and the infrared window (11 μm). Each observed radiance value is compared

with its corresponding clear-sky Composite value. Clouds are detected only when they alter the radiances by more than the uncertainty in the clear values. In this way the "threshold" for cloud detection is the magnitude of the uncertainty in the clear radiance estimates.

The ISCCP algorithm is based on the premise that the observed visible and infrared radiances are caused by only two types of conditions, *cloudy* and *clear*, and that the ranges of radiances and their variability that are associated with these two conditions do not overlap (Rossow and Garder 1993). As a result, the algorithm is based upon thresholds; a pixel is classified as cloudy only if at least one radiance value is distinct from the inferred clear value by an amount larger than the uncertainty in that clear threshold. The uncertainty can be caused both by measurement errors and by natural variability. This algorithm is constructed to be cloud-conservative, minimizing false cloud detections but missing clouds that resemble clear conditions.

The ISCCP cloud-detection algorithm consists of five steps (Rossow and Garder 1993): (1) space contrast test on a single infrared image; (2) time contrast test on three consecutive infrared images at constant diurnal phase; (3) cumulation of space/time statistics for infrared and visible images; (4) construction of clear-sky composites for infrared and visible every 5 days at each diurnal phase and location; and (5) radiance threshold for infrared and visible for each pixel.

The AVHRR Processing scheme Over cLoud Land and Ocean (APOLLO) is discussed in detail by Saunders and Kriebel (1988), Kriebel et al. (1989) and Gesell (1989). The scheme uses AVHRR channels 1 through 5 at full spatial resolution, nominally 1.1 km at nadir. The 5 spectral bandpasses are approximately 0.58-0.68 μm, 0.72-1.10 μm, 3.55-3.93 μm, 10.3-11.3 μm, and 11.5-12.5 μm. The technique is based on 5 threshold tests. A pixel is called cloudy if it is brighter or colder than a threshold, if the reflectance ratio of channels 2 to 1 is between 0.7 and 1.1, if the temperature difference between channel 4 and 5 is above a threshold, and if the spatial uniformity over ocean is greater than a threshold (Kriebel and Saunders, 1989). These tests distinguish between cloud free and cloudy pixels. A pixel is defined as cloud free if the multispectral data have values below the threshold for each test. The pixel is defined as cloud contaminated if it fails any single test, thus it is cloud conservative. Two of those tests are then used with different thresholds to identify fully cloudy pixels from the cloud contaminated ones.

The NOAA CLAVR algorithm (Phase I) uses all five channels of AVHRR (0.63, 0.86, 3.7, 11.0, 12.0 μm) to derive a global cloud mask (Stowe et al. 1991). It examines multispectral information, channel differences, and spatial differences and then employs a series of sequential decision tree tests. Cloud free, mixed (variable cloudy), and cloudy regions are identified for 2×2 global area coverage (GAC) pixel (4 km resolution) arrays. If all four pixels in the array fail all the cloud tests, then the array is labeled as cloud-free (0% cloudy). If all four pixels satisfy just one of the cloud tests, then the array is labeled as 100% cloudy. If 1 to 3 pixels satisfy a cloud test, then the array is labeled as mixed and assigned an arbitrary value of 50% cloudy. If all four pixels of a mixed or cloudy array satisfy a clear-restorer test (required for snow or ice, ocean specula reflection, and bright desert surfaces) then the pixel array is re-classified as "restored-clear" (0% cloudy). The set of cloud tests is subdivided into daytime ocean scenes, daytime land scenes, nighttime ocean scenes and nighttime land scenes.

Subsequent phases of CLAVR, now under development, will use dynamic thresholds predicted from the angular pattern observed from the clear-sky radiance statistics of the previous 9-day repeat cycle of the NOAA satellite for a mapped one degree equal area grid cell (Stowe et al. 1994). As a further modification, CLAVR will include pixel by pixel classification based upon different threshold tests to separate clear from cloud contaminated pixels and to separate cloud contaminated pixels into partial and overcast cover. Cloud contaminated pixels are radiatively "typed" as belonging to low stratus, thin cirrus, and deep convective cloud systems. A fourth type indicates all other clouds, including mixed level clouds.

Many algorithms have also been developed for cloud clearing of the TIROS-N Operational Vertical Sounder (TOVS). For example, the fifth version of the International TOVS Processing Package (ITPP-5, Smith et al. 1993), uses collocated AVHRR and HIRS/2 to cloud clear the HIRS/2 footprints. A 3×3 retrieval box of collocated AVHRR and HIRS/2 are used to determine the warm, overall and cold scenes. A scene, or HIRS/2 field of view, is classified as cloudy if any of the following conditions are meet:

1. The average AVHRR $BT_{3.7}$ or the warm signal exceeds the average AVHRR BT_{11} warm signal;
2. The skin temperature as derived from the AVHRR is more than 10°C colder than the initial guess surface temperature;

3. The average albedo for the warm HIRS/2 footprint in either of the AVHRR solar channels is greater than 25% (day tests);
4. The albedo in the HIRS visible channel (channel 20) is larger than 25% (day test);
5. The average AVHRR $BT_{3.7}$ for the warm scene is more than 4°C warmer than the skin temperature as derived by the AVHRR (night test);
6. The average HIRS/2 $BT_{3.7}$ for the warm scene is more than 4°C warmer than the skin temperature as derived by the HIRS/2 (night test); or
7. Skin temperatures derived from the AVHRRR and HIRS/2 differ by more than 2°C.

If a HIRS/2 footprint is determined cloudy, further tests are executed to determine the nature of the cloud cover. Other TOVS cloud clearing approaches (Rizzi et al. 1994) are based on the N* approach developed by Smith (1968).

Operational GOES products by NESDIS also require cloud detection and are referred to as "cloud clearing." In this application, an array of n×n contiguous pixels is categorized as either *clear*, *cloudy* or *unusable*. The *clear* arrays are subcategorized as *truly clear* and *clear/cloudy*. In an approach similar to the ITPP-5 method, clear conditions are determined based on brightness thresholds, difference thresholds, and comparison of observations with first guess profiles.

The above algorithms are noted as they have been incorporated into current global cloud climatologies. The above cloud detection schemes have been run in an operational mode over long time periods and thus faced some of the constraints of the GLI cloud mask algorithm. Many other studies (see the reference list) of cloud detection influenced this draft of the GLI mask. The GLI cloud mask algorithm builds on this work, and current work beginning developed for MODIS.

2.3 GLI Characteristics

The GLI bands used in the version 1.0 of the cloud mask algorithm are:

Band	Wavelength (μm)	Used in Cloud Mask	
8	0.55	Y	thick clouds
13	0.68	Y	clouds
19	0.88	Y	low clouds
26	1.24	Y	snow, clouds
27	1.38	Y	thin cirrus
28	1.64	Y	snow, cloud
30	3.7	Y	window
31	6.7	Y	high moisture
34	8.6	Y	mid moisture
35	10.8	Y	window
36	12.0	Y	low moisture

In preparation for a GLI day-1 cloud mask product, observations from the MODIS Airborne Simulator (MAS) (King et al. 1996), AVHRR, and the HIRS/2 are being used to develop a multispectral cloud mask algorithm. The AVHRR and HIRS/2 instruments fly on the NOAA polar orbiting satellite, while the MAS flies onboard NASA's high altitude ER-2 aircraft collecting 50 m resolution data across a 37 km swath. The multispectral nature of MAS enhances the cloud detection capability, especially for highly varying surface, atmospheric, and cloud characteristics present on the global scale.

A three month global data set of collocated AVHRR and HIRS/2 observations is also being used to develop and test the cloud masking algorithm globally. The real-time algorithm (Frey et al. 1996) collocated and analyzed the AVHRR and HIRS/2 observations and is referred to as CHAPS (Collocated HIRS/2 and AVHRR Processing Scheme). This collocated data set has the advantage of containing many IR observations that are similar to the planned GLI channels. The CHAPS data are used to assist in setting infrared thresholds for cloudy conditions over water. AVHRR GAC scenes are also being used to gain experience with global processing and the development and use of clear-sky radiance maps.

AVHRR LAC scenes are also being used to familiarize the algorithm with handling large data sets with a 1 km spatial resolution. These LAC data are also used to gain experience with incorporating satellite observations with the 1 km land/sea files and ecosystem maps.

The basic data sets currently being used to develop the cloud mask algorithm are listed below, as well as brief descriptions of the advantages and disadvantages of each. Examples are provided later.

Data Set	Advantages	Disadvantages
AVHRR LAC	Similar spatial resolution; Readily available	5 Channels; No global coverage
AVHRR GAC	Global coverage; Readily available	5 Channels; 4 km footprint
Collocated HIRS/AVHRR	Many GLI-like channels; Collocation of smaller pixels within larger footprint	Large HIRS/2 FOV; Gaps between HIRS footprints
MAS (50 channels)	Most GLI like data set; High spatial resolution	No global coverage

2.4 Cloud Mask Inputs and Outputs

The following paragraphs summarize the input and output of the GLI cloud algorithm. Details on the multispectral single field of view (FOV) and spatial variability algorithms are found in the algorithm description section. As indicated earlier, input to the cloud mask algorithm is assumed to be calibrated and navigated level 1B radiance data in channels 8, 13, 19, 26, 27, 28, 30, 31, 34, 35, and 36. Additional channels may be incorporated in the next version of the GLI cloud mask. Incomplete or bad radiometric data will create holes in the cloud mask. Additionally, the cloud mask requires several ancillary data inputs:

* sun angle, azimuthal angle, and viewing angle: obtained from MOD03 (geolocation fields),
* land/water map at 1 km resolution: currently the land/water map is provided by the EDC,
* ecosystems: 1 km map of ecosystems is desired; The Olson map of ecosystems at 10-minute resolution is used for global processing. Over North America the 1 km ecosystem provided by Tom Loveland is used,
* surface temperatures (sea and land) at best available resolution,
* snow map at the best available resolution.

The best available ancillary data will be used for the day-1 GLI cloud mask. However, it is expected that several GLI investigator products will improve upon these ancillary data, so an evolutionary development of the cloud mask is envisioned.

The output of the GLI cloud mask algorithm will be a 16 bit word for each field of view. The mask includes information about the processing path the masking took (e.g., land or ocean) and whether a view of the surface is obstructed. We recognize that a potentially large number of applications will use the cloud mask. Some algorithms will be more tolerant of cloud contamination than others. For example, some algorithms may apply a correction to account for the radiative effects of a thin cloud. In addition, certain algorithms may use spectral channels that are more sensitive to the presence of clouds than others.

The boundary between defining a pixel as cloudy or clear is sometimes ambiguous. For example, a pixel may be partly cloudy, or a pixel may appear as cloudy in one spectral channel and appear cloud free at a different wavelength. Figure 1 shows three spectral images of a

subvisual contrail taken from the MODIS Airborne Simulator (MAS). The left most panel is a MAS image in the 0.66 μm channel, a spectral channel typical of many satellites and commonly used by land surface classifications such as the NDVI. The contrail is not discernible in this image and scattering effects of the radiation may be accounted for in an appropriate atmospheric correction algorithm. The right most panel is a MAS 11 μm image (dark is cold, light is warm). Evidence of a contrail lingers though it would be difficult to justify its existence without the aid of the center panel — an image of the 1.88 μm channel. The 1.88 μm spectral channel is near a strong water vapor absorption band and, during the day, is extremely sensitive to the presence of high level clouds. While the contrail seems to have little impact on visible reflectances, its effect in the infrared window is enough to affect the retrievals of surface temperature. In this type of scene, the cloud mask needs to provide enough information to be useful to both visible and infrared applications. The 1.88 μm spectral channel on MAS is expected to exhibit similar characteristics to the 1.38 μm spectral channel to be available on GLI.

To accommodate a wide variety of applications, the mask is more than a simple yes/no decision (though bit 1 alone could be used to represent a single bit cloud mask). The cloud mask includes 4 levels of 'confidence' with regard to whether a pixel is thought to be clear (bits 1 and 2)[1] as well as the results from different spectral tests. The bit structure of the cloud mask is:

---

[1] In this document, representations of bit fields are ordered from right to left. Bit 0, or the right-most bit, is the least significant.

Figure 1. Three spectral images (0.66, 1.88 and 11 μm) taken from the MAS during the SUCCESS experiment (20 April 1996). In the 11 μm image, dark is cold.

16 BIT GLI CLOUD MASK FILE SPECIFICATION		
BIT FIELD	**DESCRIPTION KEY**	**RESULT**
0-1	Unobstructed FOV Quality Flag	00 = cloudy 01 = probably clear 10 = confident clear 11 = high confidence clear
PROCESSING PATH FLAGS		
2	Day / Night Flag	0 = Night / 1 = Day
3	Sun glint Flag	0 = Yes / 1 = No
4	Snow / Ice Background Flag	0 = Yes/ 1 = No
5-6	Land / Water Flag	00 = Water 01 = Coastal 10 = Desert 11 = Land
ADDITIONAL FLAGS		
7	Non-cloud obstruction Flag (heavy aerosol)	0 = Yes / 1 = No
8	Thin Cirrus Detected (solar)	0 = Yes / 1 = No
9	Shadow Detected	0 = Yes / 1 = No
1-km CLOUD FLAGS		
10	Result from Group I tests	0 = Yes / 1 = No
11	Result from Group II tests	0 = Yes / 1 = No
12	Result from Group III tests	0 = Yes / 1 = No
13	Result from Group IV tests	0 = Yes / 1 = No
14	Result from Group V tests	0 = Yes / 1 = No
15	Spare	o

2.4.1 Processing Path (bits 2-6)

These input bits describe the processing path taken by the cloud mask algorithm. They are important bits for the interpretation of the cloud mask results.

Bit 2 - Day / Night Flag

A combination of solar zenith angle and instrument mode (day or night mode) at the pixel latitude and longitude at the time of the observations is used to determine if a daytime or nighttime cloud masking algorithm should be applied. Daytime algorithms, which include solar reflectance data, are constrained to solar zenith angles less than 85°. If this bit is set to 1, daytime algorithms were executed.

Bit 3 - Sun glint Flag

Sun glint processing path is taken when the reflected sun angle, θ_r, lies between 0° and approximately 36°, where

$$\cos\theta_r = \sin\theta \sin\theta_o \cos\psi + \cos\theta\cos\theta_o \qquad (1)$$

where θ_o is the solar zenith angle, θ is the viewing zenith angle, and ψ is the azimuthal angle. Sun glint is also a function of surface wind and sea state.

Bit 4 - Snow / Ice Background Flag

Certain cloud detection tests (e.g., visible reflectance tests) are applied differently in the presence of snow or ice. This bit is set to a value of 0 when the cloud mask processing algorithm assumes that snow is present.

Bits 5 and 6 - Land / Water Background Flag

Bits 5 and 6 of the cloud mask output file contain information concerning the processing path taken through the algorithm. There are four possible surface type processing paths: land, water, desert, or coast. Naturally, there are times when more than one of these flags could apply to a pixel. For example, the northwest coast of the African continent could be simultaneously characterized as coast, land, and desert. In such cases, we choose to output the flag which indicates the most important characteristic for the cloud masking process. The flag precedence will be as follows: coast, desert, land or water.

Thresholds for the spectral tests are a function of surface background, land and water being the two most obvious. Therefore, each pixel will be tagged as being land or water. The 1 km United States Geological Survey (USGS) global land/water mask is currently used for this determination (available from USGS at http://edcwww.cr.usgs.gov/landdaac/1KM/1kmhomepage.html).

Some cloud detection algorithms are also ecosystem dependent. Thus, an ecosystem will be determined for each land pixel. Over North America the current version of the cloud mask uses the 1 km ecosystem map of Loveland, available from EDC. Global applications currently use the 10-minute resolution Olson World Ecosystem map.

As improved ancillary data bases become available, they will be incorporated into the GLI cloud mask. The at launch product will make use of the best available land/sea and ecosystem map with a 1 km spatial resolution.

2.4.2 Output (bits 0,1 and 7-15)

This section gives a brief description of the meaning of the output bits.

Bits 0-1 - Unobstructed FOV Quality Flag[2]

Confidence flags convey strength of conviction in the outcome of the cloud mask algorithm tests for a given FOV. When performing spectral tests, as one approaches a threshold limit, the certainty or confidence in the outcome is reduced. Therefore, a confidence flag for each individual test, based upon proximity to the threshold value, is assigned and used to work towards a final quality flag determination for the FOV. The current scheme applies a linear interpolation between a low confidence clear threshold (0% confidence of clear) and high confidence clear threshold (100% confidence clear) for each spectral test.

The final determination is a combination of the confidences of all applied tests. This determination will dictate whether additional testing (using spatial variability tests) is warranted

---

[2] There are conditions for which the cloud mask algorithm will not be executed. For example, if all the radiance values used in the cloud masking are deemed bad, then masking cannot be undertaken. Conditions (e.g., missing spectral observations) for which the cloud mask algorithm will not be executed will be set for Version 2 of the cloud mask code. Bad radiances are flagged in the data input stream. Unlike the MODIS cloud mask, an execution bit is not specified in the GLI cloud mask.

to improve the confidence. The final cloud mask determination will be clear or cloudy with a confidence level associated with it. This approach quantifies our confidence in the derived cloud mask for a given pixel.

Bit 7 - Non-cloud Obstruction

Smoke from forest fires, dust storms over deserts, and other aerosols that result in obstructing the FOV between the surface and the satellite may be flagged as "cloud." The aerosol obstruction bit will be set to on (a value of 0) if simple spectral tests indicate the possible presence of aerosols. This bit is not an aerosol product; rather, if the bit is set to zero, then the instrument may be viewing an aerosol laden atmosphere. An example of such an aerosol test is presented in Section 3.

Bit 8 - Thin Cirrus (near infrared)

GLI includes a unique spectral channel — 1.38 μm — specifically included for the detection of thin cirrus. Land and sea surface retrieval algorithms may attempt to correct the observed radiances for the effects of thin cirrus. The definition of *thin* will be determined in collaboration with the GLI Atmosphere, Land and Ocean groups. This test is discussed in Section 3 If this bit is set to 0, thin cirrus was detected using this channel.

Bit 9 - Shadow bit

Some land retrieval products are as sensitive to the presence of shadows as they are to contamination by thin clouds. If there is a need for a shadow bit, then one will be included in the GLI cloud mask. The GLI cloud masking algorithm checks for the presence of a shadow whenever bits 0 and 1 are greater than 00. Though much work remains, section 3.2.6 discusses the shadow detection algorithm. If this bit is set to zero, a shadow was detected using spectral tests. If bits 0 and 1 are 11 (low confidence clear), tests will be run to discern if a shadow from a high cloud is being cast on a lower cloud. In such a case is detected, this bit is set to zero.

Bits 10 - Thin Cirrus (infrared)

It is likely that some IR algorithms will be insensitive to thin cirrus flagged by the GLI 1.38 μm channel. In addition, this 1.38 μm channel is not available during the night. This second

thin cirrus bit indicates that IR tests detect a thin cirrus cloud. The results are independent of the results of bit 8. The definition of *thin* in the infrared will be determined in collaboration with the GLI Atmosphere, Land and Ocean groups. This test is discussed in Section 3. If this bit is set to 0, thin cirrus was detected using infrared channels.

Bits 11 through 15 - 1 km Cloud Mask

These 5 bits represent the results of tests that make use of the 1 km observations. Each group of tests is discussed in the next section. These bits are used for interpretation of the cloud mask output.

3.0 Algorithm Description

The theoretical basis of the algorithms and practical considerations are contained in this section. For nomenclature, we shall denote the satellite measured solar reflectance as R, and refer to the infrared radiance as brightness temperature (equivalent blackbody temperature using the Planck function) denoted as BT. Subscripts refer to the wavelength at which the measurement is made.

The strategy for this cloud mask algorithm is to start with single pixel (1000 m field of view) tests. Many of the single pixel tests rely on radiance (temperature) thresholds in the infrared and reflectance thresholds in the solar. These thresholds vary with surface emissivity, with atmospheric moisture and aerosol content, and with GLI viewing scan angle. A large part of the algorithm preparation will be to characterize the different situations and the different thresholds.

3.1. Confidence Flags

Most of the single pixel tests that are discussed in following sections rely on thresholds. Thresholds are never global. There are always exceptions. For example, the ratio of reflectance at 0.86 to 0.66 μm identifies cloud for values in the range $0.9 < R_{.87}/R_{.66} < 1.1$. It seems unrealistic to label a pixel with $R_{.87}/R_{.66} = 1.09$ as cloudy, and a neighboring pixel with the ratio of 1.11 as non-cloudy. Rather, as one approaches the threshold limits, the certainty or confidence in the labeling becomes more and more uncertain. An individual confidence flag is assigned to each single pixel test and is a function of how close the observation is to the thresholds. The individual confidence flags are combined to produce the final cloud mask flag for the output file (bits 1 and 2).

Each threshold determination will be pass, conditional pass, or fail along with a confidence assessment (given in a percent). Conditional pass will involve those radiances that fall within an uncertainty region of the threshold. The uncertainty will be a measure of instrument noise in that channel and the magnitude of the correction that was necessary due to surface spectral radiative properties, as well as atmospheric moisture and/or aerosol reflection contributions. The individual confidence flag will indicate a confidence level for each single

pixel test result. The initial FOV obstruction determination is an amalgamation of all confidence flags and single pixel test results This determination will dictate whether additional testing (e.g., spatial uniformity tests) is warranted to improve the confidence. The final cloud mask determination (bits 1 and 2) is a clear-sky confidence with one of four levels associated with it: definitely clear, probably clear, possibly clear and definitely obstructed. This approach quantifies our confidence in the derived cloud mask for a given pixel. This section describes the method of assigning a confidence to a given spectral test.

Many cloud detection schemes have a single threshold for a given test. For example, if the visible reflectance over the ocean is greater than 6% then the pixel is set to cloudy. The cloud masking is designed to provide information on how much confidence a user can place on the result. Each test is assigned a value between 0 and 1 representing increasing confidence in clear-sky conditions. Figure 2 is a graphical representation of how a confidence level is assigned for a spectral test. The abscissa represents the observation and the ordinate the clear-sky confidence level. In this test, an observation greater than a value of γ is determined to be a high confidence clear scene and assigned a value of 1. An observation with a value less than α is cloudy and assigned a confidence level of 0. These high confidence clear and cloud thresholds, γ and α respectively, are determined from observations and/or theoretical simulations.

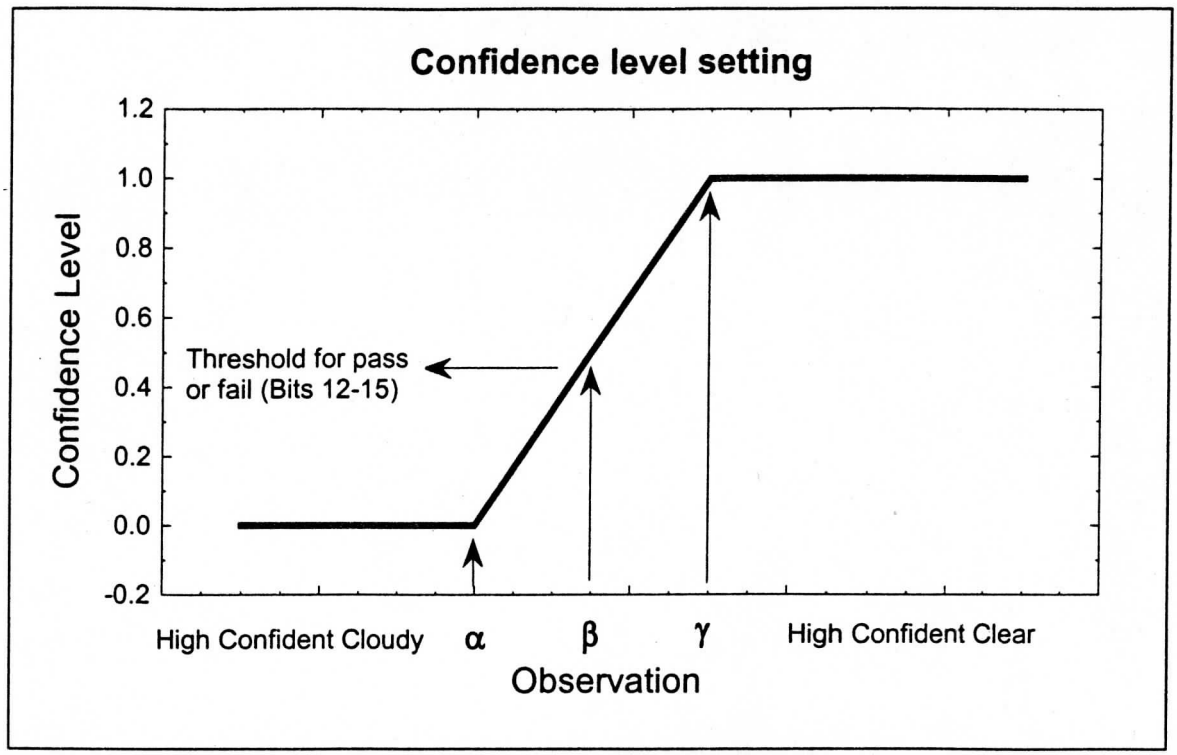

Figure 2. A graphical depiction of three thresholds used in cloud screening.

Values between α and γ are assigned a value between 0 and 1 (or 1 and 0). Assignment is based on a linear function. We have experimented with assigning confidence values based on S-functions:

$$S(x;\alpha,\beta,\gamma)=\begin{cases}0 & \text{for } x\le\alpha\\ 2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & \text{for } \alpha\le x\le\beta\\ 1-2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & \text{for } \beta\le x\le\gamma\\ 1 & \text{for } x\ge\gamma\end{cases} \tag{2}$$

The S-function is quadratic between the points α and γ. If the '2' exponent in the above equation is replaced with a 1, then the S-function becomes linear. Experiments indicate that changing between a quadratic and linear function primarily affects cloud masking at the edge of cloud systems. For simplicity we have stayed with the linear function.

In the final cloud mask only four levels of confidence are provided. Numerical values are assigned to each of these four confidence levels, 0.99 confidence of clear, 0.95 0.66 and less than 0.66 These numerical values are set based on how close the observed value is to a set of

thresholds. A description of how the final confidence level is determined is given in section 3.2.8.

3.2 *Theoretical Description of Cloud Detection*

This section discusses the physics of detecting clouds using multispectral radiances from a given field of view (FOV) or an array of FOVs, presents the application with GLI data, and indicates various problem areas.

3.2.1 Infrared Brightness Temperature Thresholds and Difference (BTD) Tests

The azimuthally averaged form of the infrared radiative transfer equation:

$$\mu \frac{\mathrm{d}I(\delta,\mu)}{\mathrm{d}\delta} = I(\delta,\mu) - (1-\omega_\circ)B(T) - \frac{\omega_\circ}{2}\int_{-1}^{1} P(\delta,\mu,\mu')\, I(\delta,\mu')\, \mathrm{d}\mu' \qquad (3)$$

In addition to atmospheric structure, which determines *B(T)*, the parameters describing the transfer of radiation through the atmosphere are the single scattering albedo, $\omega_\circ = \sigma_{sca}/\sigma_{ext}$, which ranges between 1 for a non-absorbing medium and 0 for a medium that absorbs and does not scatter energy, the optical depth, δ, and the Phase function, $P(\mu,\mu')$, which describes the direction of the scattered energy.

To gain insight on the issues of detecting clouds using IR observations from a satellite, it is useful to first consider the two-stream solution to equation (3). Using the discreet-ordinates approach (Liou, 1973; Stamnes and Swanson, 1981), the solution for the upward radiance out the top of a uniform single cloud layer is:

$$I_{obs} = M_- L_- \exp(-k\delta) + M_+ L_+ + B(T_c) \qquad (4)$$

where

$$L_+ = \frac{1}{2}\left[\frac{I\downarrow + I\uparrow - 2B(T_c)}{M_+ e^{-k\delta} + M_-} + \frac{I\downarrow + I\uparrow}{M_+ e^{-k\delta} + M_-}\right] \qquad (5)$$

$$L_{-} = \frac{1}{2}\left[\frac{I\downarrow + I\uparrow - 2B(T_c)}{M_{+}e^{-k\delta} + M_{-}} + \frac{I\downarrow - I\uparrow}{M_{+}e^{-k\delta} - M_{-}}\right] \tag{6}$$

$$M_{\pm} = \frac{1}{1 \pm k}\left(\omega_{\circ} \mp \omega_{\circ} g(1-\omega_{\circ})\frac{1}{k}\right) \tag{7}$$

$$k = \left[(1-\omega_{\circ})(1-\omega_{\circ} g)\right]^{\frac{1}{2}} \tag{8}$$

$I\downarrow$ is the downward radiance (assumed isotropic) incident on the top of the cloud layer and $I\uparrow$ is the upward radiance at the base of the layer and g is the asymmetry parameter. T_c is a representative temperature of the cloud layer.

The challenge in cloud masking is detecting thin clouds. Assuming a thin cloud layer, the effective transmittance (ratio of the radiance exiting the layer to that incident on the base) is derived from equation (4) by expanding the exponential. The effective transmittance is a function of the ratio of $I\downarrow/I\uparrow$ and $B(T_c)/I\uparrow$. Using atmospheric window regions for cloud detection minimizes the $I\downarrow/I\uparrow$ term and maximizes the $B(T_c)/I\uparrow$ term. Brightness temperature differences between the clear and cloudy sky result because of the non-linearity of the Planck function and because of the spectral variation in the single scattering properties of the cloud. The magnitude of the difference is also a function of the temperature difference between the cloud and underlying atmosphere. Observations of brightness temperature differences at two, or more, wavelengths can help separate the atmospheric signal from the cloud affect.

The anticipation is that the infrared threshold techniques will be very sensitive to thin clouds, given the appropriate characterization of surface emissivity and temperature. For example, with a surface at 300°K and a cloud at 220°K, a cloud with emissivity of 0.01 affects the sensed brightness temperature by 0.5 K. Since the expected noise equivalent temperature of GLI infrared window channel 31 is 0.05 K, the cloud detecting potential of GLI is obviously very good. The presence of a cloud modifies the spectral structure of the radiance of a clear-sky scene depending on cloud microphysical properties (e.g., particle size distribution and shape) and the temperature difference between the cloud and ground. This spectral signature is the physical reasoning behind the brightness temperature difference tests.

Simple BT threshold Test

Several infrared window threshold and temperature difference techniques have been developed. These algorithms are most effective at night for cold clouds over water and must be used with caution in other situations. The first infrared test to apply over the oceans is a simple threshold test. Over open ocean when the brightness temperature in the 11 μm (BT_{11}) channel (band 31) is less than 270 K, we assume the pixel to fail the clear-sky condition. With reference to Figure 2, the three thresholds over ocean are 267 K, 270 K and 273 K as demonstrated in Figure 3.

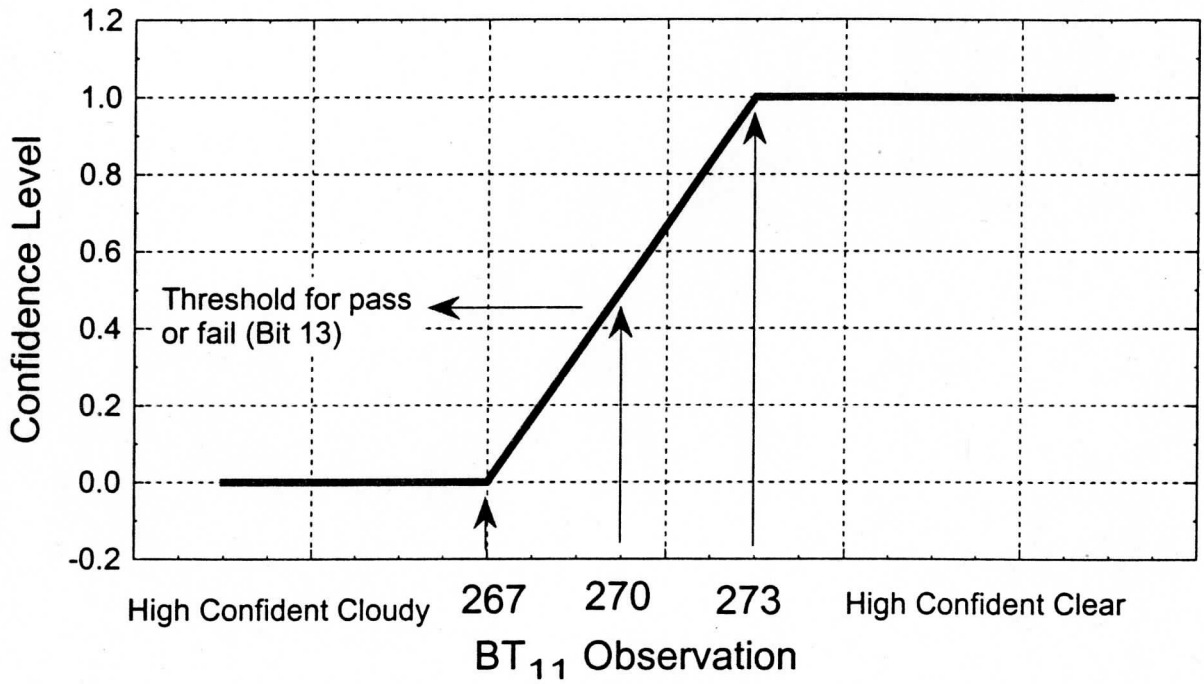

Figure 3. Thresholds for the simple IR window cold cloud test.

Cloud masking over land surface from thermal infrared bands is more difficult than over ocean due to potentially larger variations in surface emittance. Nonetheless, simple thresholds can be established over certain land features. For example, over desert regions we can expect that BT_{11}<273 K denotes cloud. Such simple thresholds will vary with ecosystem, season, and time of day; they are under investigation.

BT_{11} - BT_{12} and $BT_{8.6}$ - BT_{11} Test

As a result of the relative spectral uniformity of surface emittance in the IR, spectral tests within various atmospheric windows (such as bands 34, 35, 36 at 8.6, 10.8, and 12 μm, respectively) can be used to detect the presence of a cloud. Differences between BT_{11} and BT_{12} are widely used for cloud screening with AVHRR measurements, and this technique is often referred to as the split window technique. Saunders and Kriebel (1988) used BT_{11}-BT_{12} differences to detect cirrus clouds--brightness temperature differences are greater over clouds than clear conditions. Cloud thresholds were set as a function of satellite zenith angle and the BT_{11} brightness temperature. Inoue (1987) also used BT_{11}-BT_{12} versus BT_{11} to separate clear from cloudy conditions.

In difference techniques, the measured radiances at two wavelengths are converted to brightness temperatures and subtracted. Because of the wavelength dependence on optical thickness and the non-linear nature of the Planck function (B_λ), the two brightness temperatures are often different.

The basis of the split window and tri-spectral technique for cloud detection lies in the differential water vapor absorption that exists between different window channel (8.6 and 11 μm and 11 and 12 μm) bandwidths. These spectral regions are considered to be part of the atmospheric window, where absorption is relatively weak. Most of the absorption lines are a result of water vapor molecules, with a minimum occurring around 11 μm. Since the absorption is weak, BT_{11} can be corrected for moisture absorption by adding the scaled brightness temperature difference of two spectrally close channels with different water vapor absorption coefficients; the scaling coefficient is a function of the differential water vapor absorption between the two channels. This is the basis for sea surface temperature retrieval.

With a reasonable estimate of the sea surface temperature and total precipitable water one can develop appropriate thresholds for cloudy sky detection. For example;

$$BT_{11} + a_{PW} * (BT_{11}-BT_{12}) < SST \quad (9)$$

or

$$BT_{11} + b_{PW} * (BT_{11}-BT_{8.6}) < SST \quad (10)$$

where a_{PW} and b_{PW} are determined from a lookup table as a function of total precipitable water vapor (PW). This approach has been used operationally for 4 years using 8.6 and 11 μm

bandwidths from the NOAA-10 and NOAA-12 and the 11 and 12 μm bandwidths from the NOAA-11, with a coefficient independent of PW (Menzel et al. 1993, Wylie et al. 1994).

To demonstrate this technique with observations, a global data set of collocated AVHRR GAC 11 and 12 μm and HIRS 8.6 and 11 μm scenes were collected and the total column PW was estimated from integrated model mixing ratios to determine a direct regression between PW and the split window thresholds. The regressions were then modified for use with MODIS Airborne Simulator (MAS) bandwidths in TOGA/COARE data sets. Simulations and observations have shown the regression slopes to be consistent with those found for the AHVRR/HIRS clear scenes; however, the intercept values needed adjustment.

A disadvantage of the split window brightness temperature difference (BTD) approach is that water vapor absorption across the window is not linearly dependent on PW, thus second order relationships are sometimes used. The GLI has a unique capability with the measurements at three wavelengths in the window, 8.6, 10.8, and 12 μm. The three spectral regions mentioned are very useful in determination of cloud free atmospheres. Because the index of refraction varies quite markedly over this spectral region for water, ice, and minerals common to many naturally occurring aerosols, the effect on the brightness temperature of each of the spectral regions is different, depending on the absorbing constituent.

A tri-spectral combination of observations at 8.6, 11 and 12 μm was suggested for detecting cloud and cloud properties by Ackerman et al. (1990). Strabala et al. (1994) further explored this technique by utilizing very high spatial-resolution data from the MODIS Airborne Simulator (MAS). The physical premise of the technique is that ice and water vapor absorption peak in opposite halves of the window region; so that positive 8.6 minus 11 μm brightness temperature differences indicate cloud while negative differences, over oceans, indicate clear regions. The relationship between the two brightness temperature differences and clear-sky have also been examined using collocated HIRS and AVHRR GAC global ocean data sets. As the atmospheric moisture increases, $BT_{8.6}$-BT_{11} decreases while BT_{11}-BT_{12} increases.

Based on these observations, a threshold is set for clear-sky conditions. The clear-sky threshold is set for both differences:

$$T8M11=[-1.64805 \ln (PW)] - 3.19767 \quad (11)$$

$$T11M12= (0.488198 PW) - 0.456924 \quad (12)$$

If $BT_{8.6}$-BT_{11} > T8M11 and BT_{11}-BT_{12} > T11M12, then a cloud is assumed.
High confidence clear conditions are

$$BT_{8.6}\text{-}BT_{11} < T8M11 - 0.5 \text{ and } BT_{11}\text{-}BT_{12} > T11M12 - 0.5 \quad (13)$$

and low confidence clear condition is

$$BT_{8.6}\text{-}BT_{11} < T8M11 + 0.5 \text{ and } BT_{11}\text{-}BT_{12} > T11M12 + 0.5 \quad (14)$$

The above conditions assume an estimate of the precipitable water is available. These equations demonstrate that a relationship between T8M11 and T11M12 exists and thus a PW value is not needed, but rather given a value of T8M11, to be a clear pixel requires T11M12 to fall within a certain range of values.

Brightness temperature difference testing can also be applied over land with careful consideration of variation in spectral emittance. For example, BT_{11}-$BT_{8.6}$ has large negative values over daytime desert and is driven to positive differences in the presence of cirrus. Some land regions have an advantage over the ocean regions because of the larger number of surface observations, which include air temperature, and vertical profiles of moisture and temperature. Work on developing brightness temperature difference tests for application over land scenes continues with the MAS observations.

BT_{11} - $BT_{3.7}$ Test

GLI band 30 (3.7 μm) measures radiances in another window region near 3.5-4 μm so that the difference between BT_{11} and $BT_{3.7}$ can also be used to detect the presence of clouds. At night the difference between the brightness temperatures measured in the shortwave (3.7 μm) and in the longwave (11 μm) window regions (BT_{11}-$BT_{3.7}$) can be used to detect partial cloud or thin cloud within the GLI field of view. Small or negative differences are observed only for the case where an opaque scene (such as thick cloud or the surface) fills the field of view of the sensor. Negative differences occur at night over extended clouds due to the lower cloud emissivity at 3.7 μm.

During the daylight hours the difference between BT_{11} and $BT_{3.7}$ becomes large negative because of reflection of solar energy at 3.7 μm. This technique is very successful at detecting low level water clouds. BT_{11}-$BT_{3.7}$ are not applied over deserts during daytime, as bright desert

regions with highly variable emissivities tend to be classified incorrectly as cloudy with this test. Experience indicates that the actual thresholds will need to be adjusted for ecosystem type.

Moderate to large differences between BT_{11} and $BT_{3.7}$ result when a non-uniform scene (e.g., broken cloud) is observed. The different spectral responses to a scene of non-uniform temperature is a result of Planck's law. The brightness temperature dependence on the warmer portion of the scene increasing with decreasing wavelength (the shortwave window Planck radiance is proportional to temperature to the thirteenth power, while the longwave dependence is to the fourth power). Differences in the brightness temperatures of the longwave and shortwave channels are small when viewing mostly clear or mostly cloudy scenes; however, for intermediate situations the differences become large (greater than 3°C). The following table lists the thresholds used, which are based on previous studies and our own investigations.

11 - 3.7 μm Low Cloud Test

Scene Type	Threshold	High confidence clear	Low confidence clear
Day ocean	-8.0K	-6.0K	-10.0K
Night ocean	0.60K	0.50K	0.70K
Day land	-12.0K	-10.0K	-14.0K
Night land	0.60K	0.50K	0.70K
Day polar	-9.0K	-7.0K	-11.0K
Night polar	0.60K	0.50K	0.70K
Desert	-18.0, -3K	>-16, <-5K	<-20, >-1K

Infrared window tests at high latitudes are difficult. Distinguishing clear and cloud regions from satellite IR radiances is a challenging problem due to the cold surface temperatures. Yamanouchi et al. (1987) describe a nighttime polar (Antarctic) cloud/surface discrimination algorithm based upon brightness temperature differences between the AVHRR 3.7 and 11 μm channels and between the 11 and 12 μm channels. Their cloud/surface discrimination algorithm was more effective over water surfaces than over inland snow-covered surfaces. A number of problems arose over inland snow-covered surfaces. First, the temperature contrast between the cloud and snow surface became especially small, leading to a small brightness temperature

difference between the two infrared channels. Second, the AVHRR channels are not well-calibrated at extremely cold temperatures (< 200 K). This latter condition will be minimized using GLI.

$BT_{3.7}$ - BT_{12} Test

This window brightness temperature difference test is applied during the nighttime. This difference is useful for separating thin cirrus and cloud free condition and is relatively insensitive to the amount of water vapor in the atmosphere (Hutchison and Hardy, 1995).

$BT_{6.7}$ and BT_{11} - $BT_{6.7}$

Detection of clouds over polar regions during winter is difficult. Under clear-sky conditions, strong surface radiative temperature inversions often exist over polar regions during winter. Thus, IR channels whose weighting function peaks low in the atmosphere will often have a larger BT than a window channel. For example, $BT_{8.6} > BT_{11}$ in the presence of an inversion. The surface inversion can also be confused with thick cirrus cloud; this can be mitigated by other tests (e.g., the magnitude of BT_{11} or the BT_{11}-BT_{12}). Analysis of BT_{11}-$BT_{6.7}$ has shown large negative difference in winter time over the Antarctic Plateau and Greenland, which may be indicative of a strong surface inversion and thus clear skies (Ackerman 1996). In clear-sky situations, the 6.7 μm radiation measured by satellite instruments is emitted by water vapor in the atmospheric layer between approximately 200 and 500 hPa (Soden and Bretherton 1993; Wu et al. 1993) and has a brightness temperature ($BT_{6.7}$) related to the temperature and moisture in that layer. The 6.7-μm radiation emitted by the surface or low clouds is absorbed in the atmosphere and is not sensed by the satellite instruments. On the other hand, absorption by atmospheric gases is weak at 11-μm. Under clear-sky conditions, the measured 11 μm radiation originates primarily at the surface, with a small contribution by the near-surface atmosphere. Because the surface is normally warmer than the upper troposphere, BT_{11} is normally warmer than 6.7-μm brightness temperature; thus the difference, BT_{11}-$BT_{6.7}$, is normally greater than zero. Large negative differences in BT_{11}-$BT_{6.7}$ (less than -10°K) exist over the Antarctic Plateau and Greenland during their respective winters and are indicative of clear-sky conditions and the existence of strong low-level temperature inversions.

In polar regions, strong surface radiation inversions can develop as a result of longwave energy loss at the surface due to clear-skies and a dry atmosphere. Temperatures over Antarctica near the surface can reach 200 K (Stearns et al. 1993), while the middle troposphere is ~235K. Under such conditions, satellite channels located in strong water vapor absorption bands, such as the 6.7-μm channel, will have a warmer equivalent brightness temperature than the 11-μm window channel. This brightness temperature difference between 11- and 6.7-μm is an asset to detecting cloud-free conditions over elevated surfaces in the polar night (Ackerman 1996). Clouds inhibit the formation of the inversion and obscure the inversion from satellite detection if the IWP is greater than approximately 20 g m^{-2}. A positive difference of BT_{11}-$BT_{6.7}$ is a cloud; differences smaller than -10C are representative of cloud. This test is applied only during the polar winter.

3.2.2 Non-cloud obstruction flag

A heavy aerosol laden atmosphere may result in a low confidence clear scene. Certain simple tests may be constructed that can indicate that the FOV is contaminated with an aerosol and not a cloud. For example, negative values of BT_{11}-BT_{12} are often observed over deserts and can be attributed to the presence of dust storms (Ackerman 1996). Under such conditions, provided BT_{11} is warm, the non-cloud obstruction bit (Bit 7) would be set. The tri-spectral technique may also be used to flag a region as potentially contaminated with volcanic aerosol. These tests are currently under implementation. In addition, this bit may be used to pass information on to the aerosol retrieval algorithm. For example, over oceans it might indicate the passing or failing of a uniformity test.

3.2.3 Near Infrared 1.38 μm Cirrus Test

This is a new approach to cirrus detection suggested by the work of Gao et al. (1993) and is still under development. The GLI band 27 (1.38 μm) will use reflectance thresholds on a per pixel basis to detect the presence of thin cirrus cloud in the upper troposphere under daytime viewing conditions. The strength of this cloud detection channel lies in the strong water vapor absorption in the 1.38 μm region. With sufficient atmospheric water vapor present (estimated to be about 0.4 cm precipitable water) in the beam path, no upwelling reflected radiance from the

earth's surface reaches the satellite. Since 0.4 cm is a small atmospheric water content, most of the earth's surface will indeed be obscured in this channel. With relatively little of the atmosphere's moisture located high in the troposphere, high clouds appear bright; reflectance from low and mid level clouds is partially attenuated by water vapor absorption.

Simple low and high reflectance (normalized by incoming solar at the top of the atmosphere) thresholds will be used to separate thin cirrus from clear and thick (near infrared cloud optical depth > ~ 0.2) cloud scenes. These thresholds will be set initially using a multiple-scattering model with the assumption of no surface reflectance contribution to the satellite observed radiance, i.e., a dark background. Ben-Dor (1994) analyzed a scene from the AVIRIS to demonstrate that thin cirrus detection using 1.38 μm observations may depend on surface elevation, dry atmospheric conditions, and high albedo surface. New injections of volcanic aerosols into the stratosphere may also impact this test.

The MAS 1.88 μm channel is being used as a surrogate to GLI channel 27 to gain experience with defining the thin cirrus bit. If the reflectance lies between the clear-sky threshold and below a thick cloud, then the cirrus bit will be set to 0. We subjectively define a thin cirrus as a cloud that has a small impact on the visible reflectance, enabling atmospheric correction to be applied to retrieve land surface properties (i.e., NDVI). Figure 1 demonstrates the potential utility of the 1.38 μm channel. The three panels are of observations made from the MAS during the SUCCESS field campaign. The MAS has a 1.88 μm channel which, like the 1.38 μm channel is near a strong water vapor absorption band. The panel to the left is a visible image that has been histogram normalized. The right panel is an 11 μm image where dark regions represent cold scenes. The center scene is the 1.88 μm image which clearly shows the presence of contrails that are difficult to see in the 11 μm image and indiscernible in the 0.66 μm image. Given the sensitivity to thin high clouds, the new 1.38 μm channel may detect a much larger cloud coverage than previous satellite algorithms have indicated.

The initial 1.38 μm reflectance thresholds will be based on theoretical simulations and MODIS observations, and will be adjusted using inflight data after GLI is launched.

3.2.4 Infrared Thin Cirrus Test

This second thin cirrus bit indicates that IR tests detect a thin cirrus cloud. This test is independent of the thin cirrus flagged by the GLI 1.38 μm channel. This test will apply brightness temperature differences tests to detect the presence of thin cirrus. The APOLLO scheme tests for the presence of thin cirrus using the split window analysis. Analysis of BT_{11}-BT_{12} and BT_8-BT_{11} is also effective in detecting the presence of thin cirrus clouds.

3.2.5 Detection of Cloud Shadows

The detection of cloud shadows is a problem that has not been addressed adequately in the literature. Clear-sky scenes that are potentially affected by shadows can be theoretically computed given the viewing geometry, solar azimuth and zenith angles, cloud edges distribution and cloud altitude. This approach requires too much CPU to run operationally, and all the information (e.g., cloud altitude) is not available to the cloud mask algorithm. Therefore, as with clouds, solar reflectance tests will be explored for a cloud shadow detection algorithm. Further work in this area has been initiated.

Currently, the cloud masking algorithm checks for shadows whenever a high confident clear scene is identified. Shadow detection is based on reflectance at 1.05, 0.88 and 0.66 μm. A cloud shadow algorithm will be developed, otherwise this bit will be replaced with a cloud detection test.

3.2.6 Visible Reflectance Tests

This is a single channel test whose strength is discriminating bright clouds over dark surfaces (e.g., stratus over ocean) and weakness is clouds over bright surfaces (e.g., snow). Two different channels are used in this test dependent on the ecosystem. The 0.66 μm (band 1) is used over oceans, land and polar region. The 0.88 μm reflectance test is also applied over polar and desert scenes. The nominal thresholds are given below.

0.66 μm Reflectance Threshold Test			
Scene Type	Threshold	High confidence clear	Low confidence clear

Day ocean	7.0%	6.5%	8.0%
Day land	16.0%	14.0%	18.0%
Day polar	20.0%	22.0%	18.0%
0.88 μm Reflectance Threshold Test			
Scene Type	Threshold	High confidence	Low confidence
Day polar	10.0%	12.0%	8.0%
Desert	30.0%	26.0%	34.0%

These thresholds are being set based on observations from the AVHRR and MAS. The reflectance test is angular dependent and is also applied in sun glint regions as identified by the sun glint test.

3.2.7 Reflectance Ratio Test

The reflectance ratio test uses channel 19 divided by channel 13 ($r_{.87}/r_{.68}$). This test makes use of the fact that the spectral reflectance at these two wavelengths is similar over clouds (ratio is near 1) and different over water and vegetation. Using AVHRR data this ratio has been found to be between 0.9 and 1.1 in cloudy regions. If the ratio falls within this range, cloud is indicated. New analyses (McClain 1993) suggest that the minimum value may need to be lowered to about 0.8, at least for some cases. For cloud-free ocean the ratio is expected to be less than 0.75 (Saunders and Kriebel 1988). This ratio is generally observed to be greater than 1 over vegetation. Adjustments to the ratio thresholds will be made as necessary for GLI data and must be a function of the ecosystem.

Figure 4 illustrates some of the complexities of desert ecosystems as demonstrated by the visible ratio. The observations are from the AVHRR on the NOAA-9 and are over the Arabian Sea, the Arabian Peninsula, and surrounding regions. The figure shows histograms of ratio values for coastal/water scenes, as well as desert and more densely vegetated areas in the Persian Gulf region from approximately 15-25° N latitude and 50-70° E longitude. Almost all of the observations recorded in the histograms were from clear-sky conditions, as determined by

inspection of visible and IR imagery. As suggested by the histograms of $r_{.87}/r_{.66}$, clear-sky ocean scenes have a ratio of less than 0.75. The surface type classifications are from the Olson World Ecosystems data set. One can immediately see that clear-sky desert values of the visible ratio cover a large range and where that range would indicate cloudy skies for vegetated surfaces. Also note the large amount of overlap between the desert and shrub/grassland categories. This figure shows that clear-sky spectral threshold tests need to be applied very carefully in arid regions and also points out the need for high-resolution ecosystem maps. This test will not be performed over desert, semi-desert and some agricultural ecosystems.

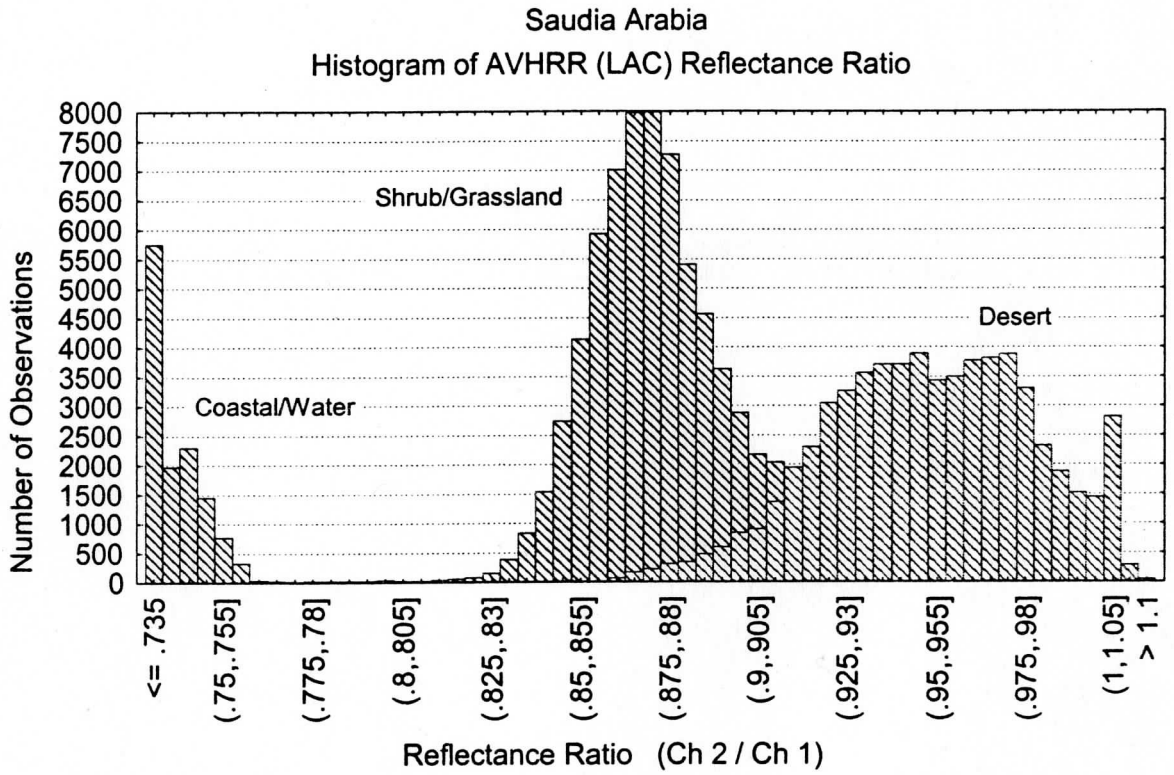

Figure 4. Histogram of the frequency of occurrence of the AVHRR Ch2/Ch1 ratio for a scene over the Arabian peninsula and Arabian Sea.

This ratio test is always performed over water during the day. When applied in regions of possible sun glint, an angular dependency is included in the thresholds. These thresholds are based on analysis of AVHRR LAC and GAC data and the APOLLO algorithm. A histogram analysis of the visible ratio from AVHRR data was performed for 1° increments of reflected sun angle, θ_r (equation 1).

Mixed land and water fields-of-view produce a ratio that is similar to cloud. Thus, coastal regions may be flagged as cloudy.

3.2.8 Confidence Flags

Each of the tests above returns a confidence level ranging from 1 (high confidence that the pixel is clear) to 0 (high confidence that the pixel is cloudy). The individual confidence levels must be combined to determine a final decision on clear or cloudy. We shall denote the confidence level of an individual test as F_i and the final quality flag as Q. There are different methods of combining these individual tests to yield the final quality flag (bits 0 and 1). We have experimented with a variety of methods of combining the confidence of individual tests. The primary effect occurs on the boundaries of the cloud system.

Several tests are not independent of one another. For example, consider daytime over oceans in regions without sun glint. If stratocumulus clouds are present, they will likely be detected by the visible reflectance test, the reflectance ratio test and the BT_{11}-$BT_{3.7}$. These same tests will likely miss the presence of thin uniform cirrus clouds, which would probably be detected by the tri-spectral tests (combinations of BT_8, BT_{11}, and BT_{12}). Very thin cirrus clouds would best be detected by the 1.38 and 13.9 μm micron tests, two tests which have difficulty detecting low level clouds. Because of this overlap in the type of clouds different tests detect, each test is considered in one of four groups. The five groups are:

Group I (Simple IR threshold test)

BT_{11}

$BT_{6.7}$

Group II (Brightness temperature difference)

Tri-spectral test (BT_8 - BT_{11} and BT_{11}- BT_{12})

BT_{11}- $BT_{3.7}$

BT_{11}- $BT_{6.7}$

Group III (Solar reflectance tests)

$r_{0.87}$

$r_{.87}/r_{.66}$

Group IV (NIR thin cirrus)

$r_{1.38}$

Group V (IR thin cirrus)

BT_{11}- BT_{12}

BT_{12}- $BT_{3.7}$

It is likely that the number of these groups will expand in version 2 of the cloud mask. A minimum confidence is determined for each group,

$$G_{i=1,4} = \min[F_i] \tag{15}$$

The final cloud mask is then determined from the product of the results from each group;

$$Q = \sqrt[N]{\prod_{i=1}^{N} G_i} \tag{16}$$

This approach is still clear-sky conservative. If any test is highly confident that the scene is cloudy (F_i=0), the final cloud mask is 00.

The algorithm is divided into eight conceptual domains according to surface type and solar illumination:

1. daytime land surface,
2. daytime water,
3. nighttime land,
4. nighttime water,
5. daytime desert (currently daytime only),
6. nighttime desert (currently daytime only),
7. daytime snow covered regions, and
8. nighttime snow covered regions.

"Daytime" is defined as a solar zenith angle < 85° (and the instrument is in daytime mode). The "desert" classification is based on the 10-minute Olson World Ecosystems data set. A USGS (United States Geological Survey) 1 km land/sea tag file is used for land/water discrimination. For all observations within a given domain, it is generally expected that: 1) the same tests may be performed, and 2) threshold values for each of these tests will not change. Tests defined for nighttime polar or desert observations are still under development. The default nighttime land

algorithm is substituted in both cases. It is expected that more domains will be established in the future.

3.2.9 Radiance Spatial Uniformity

The infrared window spatial uniformity test (applied on 3 by 3 pixel segments) is effective over water and is to be used with caution in other situations. Most ocean regions are well suited for spatial uniformity tests; such tests may be applied with less confidence in coastal regions or regions with large temperature gradients (e.g., the Gulf Stream). Various spatial tests exist such as the spatial coherence and the ISCCP space contrast test.

The spatial coherence test (Coakley and Bretherton, 1982) is based on the assumption of a uniform background and single-layered, optically thick cloud systems. The method is based upon the computation of the mean and standard deviation for a group of pixels using 11 μm radiances. When the standard deviation is plotted versus the mean, an arch shaped structure is often observed. The warm pixels with low values of standard deviation are assumed to be clear regions. The clear-sky FOVs can be selected as those within a standard deviation threshold (which is fixed at a small value) of the warm foot of the arch. Note that the derived clear-sky foot of the arch should have a temperature consistent with the thresholds derived using the individual FOV tests.

The ISSCP space contrast (sc) test, described in Rossow and Garder (1993), is similar to that of spatial coherence and is physically based upon the fact that clear pixels tend to be warmer than cloudy pixels and exhibit less spatial variability. First, for a small local region the pixel with the largest brightness temperature (BT_{11}^{max}) is found. All pixels colder than the spatial contrast ($BT_{11}^{max} - \Delta_{sc}$) value are labeled as cloudy; all others, including the warmest pixel are labeled as undecided. The size of the contrast threshold must be larger than the magnitude of natural variation at the surface and smaller than that caused by clouds. Values of Δ_{SC} near 3.5°C are common over ocean regions. It is important that the class of pixels be identical and that the size of the region be chosen carefully. When extending into coastal regions and land regions containing mixed land and water, pixels are excluded from this test since the inherent contrast between land and water surface radiances would dominate the results. For regions that are too large, there is increased likelihood of spatial variations in surface parameters. The shape of the

test regions can also be important since meridional gradients in surface temperature generally are larger than zonal gradients.

Similar spatial tests will be incorporated into the GLI cloud mask for uncertain obstructed FOVs. The specific test will likely depend on how the data stream is structured regarding the ease of compositing pixels over geographic regions. Uniform stratus can also give the appearance of a uniform ocean, thus the spatial tests must often complement the threshold tests (e.g., the tri-spectral test).

The GLI cloud mask currently uses a spatial variability test over oceans and large lakes. The tests are used to modify the confidence of a pixel being clear. If the confidence flag of a pixel is < 0.95, the variability test is implemented. If the difference between the pixel of interest and any of the surrounding pixel brightness temperatures is $>0.5°C$, the scene is considered variable and the confidence lowered.

Surface temperature variability, both spatial and temporal, is larger over land than ocean, making land scene spatial uniformity tests difficult. GLI spatial uniformity tests over land will be constrained to similar ecosystems in a given geographic regime. These tests have yet to be developed.

The reflectance uniformity test (similar to CLAVR) is applied by computing the maximum and minimum values of GLI 0.66 μm or 0.87 μm reflectances within a 3 x 3 pixel array. Pixel arrays with band 1 reflectance differences greater than threshold 1 (around 9%) over land or band 2 reflectance differences greater than threshold 2 (possibly 0.3%) over ocean are labeled in CLAVR as mixed (Stowe et al. 1993). The value over ocean is low because a cloud-free ocean is almost uniformly reflective, while non-uniformity is assumed to be caused by cloudiness. This test will be refined for GLI applications.

3.2.10 Clear-Sky Radiance Composite Maps

Composite maps have been found to be very useful by ISSCP. The MODIS cloud mask relies on composite maps, but to a lesser extent than ISCCP since the advantages of higher spatial resolution and more spectral bands will change the application and the need. The disadvantage of composite maps is the large memory and CPU required for effective application. To reduce storage and CPU requirements, we are not planning on making use of clear-sky radiance composite maps for the GLI cloud mask algorithm.

4.0 Practical Application of Cloud Detection Algorithms

To review:

The algorithm is divided into eight conceptual domains according to surface type and solar illumination: daytime land surface, daytime water, nighttime land, nighttime water, daytime desert, nighttime desert, and daytime and nighttime polar regions. "Daytime" is defined as a solar zenith angle < 85° while "polar" currently refers to any region poleward of 60° latitude. The "desert" classification is based on the 10-minute Olson World Ecosystems data set. A USGS (United States Geological Survey) 1 km land/sea tag file is used for land/water discrimination. For all observations within a given domain, it is generally expected that: 1) the same tests may be performed, and 2) threshold values for each of these tests will not change. As yet, there are no sets of tests defined for nighttime polar or desert observations. The default nighttime land algorithm is substituted in both cases. It is expected that more domains will be established in the future.

Once a pixel has been assigned to a particular domain, the spectral data corresponding to that location are subjected to a series of threshold tests designed to detect the presence of clouds in the instrument FOV. These tests are the heart of the cloud mask algorithm. There are several types of tests, none of which are equally good at detecting all cloud conditions. For example, some are more reliable for low cloud or thin cirrus cloud detection, while others are better at finding any kind of optically thick cloud. Accordingly, tests which indicate similar cloud conditions are grouped together to obtain several intermediate results that are then combined to form a final cloud mask value. The tests are grouped so that independence between them is maximized. All tests which detect thin cirrus might make up a group, for example, while those which find high, cold clouds would form another, and low-level cloud detection tests could make a third set. Of course, few if any spectral tests are completely independent of all the others.

The result of each spectral threshold test is expressed as a "confidence" which indicates the strength of the observed radiance signature compared to that which is expected for the cloud condition in question. For example, one very fundamental test performed for water surfaces is the "cold cloud test." Over open waters any scene with an observed 11 μm brightness temperature colder than ~270°K must be at least partially cloudy. Therefore, the threshold value for this test is

set at that value. The actual cutoff temperature for any given FOV varies slightly, however, because of differing amounts of atmospheric water vapor attenuation due to changes in actual water vapor content or instrument viewing angle. Consequently, a "confidence window" is constructed with boundaries at 267° and 273°K. Since the ultimate goal of the algorithm is to specify a confidence of clear-sky, an observed brightness temperature of < 267°K is defined to have 0 confidence while a measurement of > 273°K has 1.00 confidence. *Note that this confidence does not yet refer to clear-sky conditions, only to the particular condition tested for.* In this case it indicates only the extent of confidence that the FOV did *not* contain significant amounts of opaque, cold clouds. Observations between 267° and 273°K result in confidences ranging linearly between 0 and 100%. Figure 3 showed an example of the relationship between test thresholds and confidence boundaries. Note that the threshold value corresponds to a confidence of 50%.

When all tests within a group have been performed, the minimum resulting confidence from among them is taken to be representative of that group. Then the other groups of tests are performed with group confidences determined in the same way. These confidences indicate absence of particular cloud types. A final step is to combine the group confidences, assumed to be independent, by multiplying them together. Only at this point does the confidence value reflect the surety of clear-sky conditions.

Using this algorithm, most observations have either high confidences (> 0.95) or very low confidences (< 0.05) of unobstructed views of the surface. There are always those difficult scenes, however, which result in intermediate values of the cloud mask. These tend to be found at cloud boundaries, or where low clouds are found over water surfaces at night, or over certain land surfaces such as desert or other sparsely-vegetated regions. In these cases (final confidence > 0.05 and <0.95), spatial and/or temporal continuity tests are conducted. Currently, only spatial continuity testing has been implemented and this only for water surfaces. 11 μm brightness temperature differences between the pixel of interest and the surrounding eight are checked for consistency. If all the differences are less than 0.5 K, the confidence is adjusted upward by one "level" (e.g. > 0.66 to > 0.95).

4.1 Ancillary Data Set Requirements

A number of preprocessing steps will be made before the cloud masking algorithm is applied. First, each pixel in the scene will be tagged as being land or water, and if land, a land/water percentage. Second, each land pixel will be classified as to its ecosystem and its elevation will be designated as relatively flat, valley, isolated mountainous region, low mountains or hills, generally mountainous, or extremely rugged mountains.

Information on surface temperature and sea state will be sought from surface observations, Reynolds blended analysis, and NMC model 3-hour surface analyses of temperature.

4.2 Implementation of the Cloud Mask Algorithms

4.2.1 Outline of cloud mask algorithm

The hierarchical approach used in version 1 of the cloud mask is:

(1) Determine if the pixel is of a land or water scene.

(2) Determine the ecosystem type.

(3) Determine if pixel is in a sun glint region.

(4) Determine if the pixel is in a day or night regime.

(5) Retrieve information from snow cover and ice data base.

(6) Apply appropriate single FOV masking tests and set initial unobstructed FOV determination for the given domain. Initial confidence flag is assigned for each test result, depending on its relative position to the threshold (see Section 3).

- For daytime testing, solar zenith angles are constrained to be less than 85°.
- Ocean tests are applied to open ocean and for large lakes.
- Sun glint occurs when the reflected sun angle lies between 0° and 36°.
- The land algorithm is applied to non-desert and non-water areas.
- The desert algorithm is applied to desert ecosystems.
- The snow algorithm is applied to regions passing the NSDI test.

- For the single pixel clear-sky determination, 11 single FOV tests are implemented and an obstructed/not obstructed bit set (0 for obstructed, 1 for clear) for each test.

(7) The single FOV cloud test results are grouped and the minimum of each group determined.

(8) The group minimums are then multiplied together, and the N^{th} root taken (where N represents the number of groups) producing the initial cloud mask (Section 3.2.8). If any of the individual tests are high confidence cloudy (clear confidence of 0), the product is zero.

(9) If confidence level is still uncertain (between 0.05 and < 0.95), use spatial uniformity tests on 3×3 pixel regions (Currently not implemented over land).

- Spatial IR variability test applied with band 31 using $\Delta_{SV} = 0.50$ K over water.
- Adjust quality flag if appropriate by increasing or decreasing confidence levels.

(10) Output cloud mask.

4.2.2 Cloud mask examples

The cloud mask algorithm has been applied to several data sets selected by the cloud mask group, as well as scenes selected by other science team members. This section includes several examples of the output from the current version of the GLI cloud mask. It is not intended as a validation section, but rather as a visual indication of the success and problems of the current version.

AVHRR cloud mask Data sets

Figure 5 is a three panel image of NOAA-14 AVHRR GAC observations. The left most panel is the channel 1 (0.6 μm) image and the middle panel the channel 4 (11 μm) image. The scene is over the tropical Pacific Ocean. The resulting cloud mask output file is represented in the right most panel. The legend is as follows

Black: High confident clear (bits 1-2 equal 11)

Dark Gray: Probably clear (bits 1-2 equal 10)

Light Gray: Maybe Clear(bits 1-2 equal 01)

White: High confident obstructed FOV (bits 1-2 equal 00)

Notice that the confidence in clear-sky scene tends to drop over the sun glint region, as it is difficult to detect clouds in areas affected by sun glint. The additional channels of GLI should improve our clear-sky detection capabilities in sun glint regions.

Another 3 panel image of NOAA-14 AVHRR GAC observations is shown in Figure 6. This scene includes parts of North Africa and the Atlantic Ocean. The *blocky* structure of the cloud mask over land results from the 10-minute ecosystem map; land spectral tests are being applied to desert regions. This points to the need for an accurate global 1 km ecosystem map, one that is consistent with the land/sea map. The high confidence obstructed scenes over the Saharan region is a dust storm. Most of this dust storm has negative values of BT_{11}-BT_{12}, setting the aerosol obstruction bit to zero (bit 8 in the mask output file).

Future AVHRR Data Processing

An entire month of AVHRR GAC data has been obtained from the AVHRR Pathfinder Program. This data set is being processed to gain further experience with implementing clear-sky

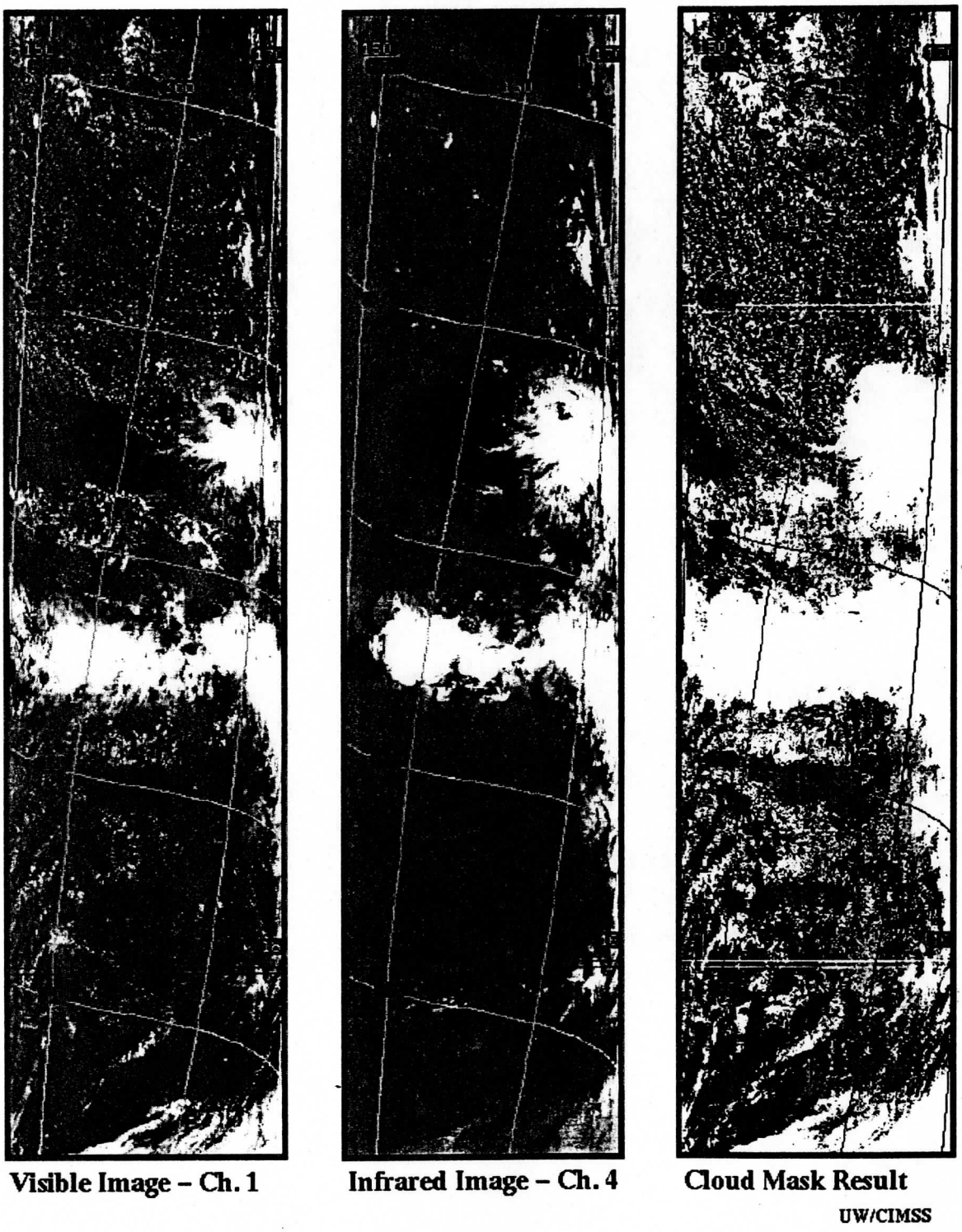

Figure 5. An example of the cloud mask derived from AVHRR data for a scene over the Pacific Ocean (right hand panel), together with images from two of the AVHRR bands used in generating the mask.

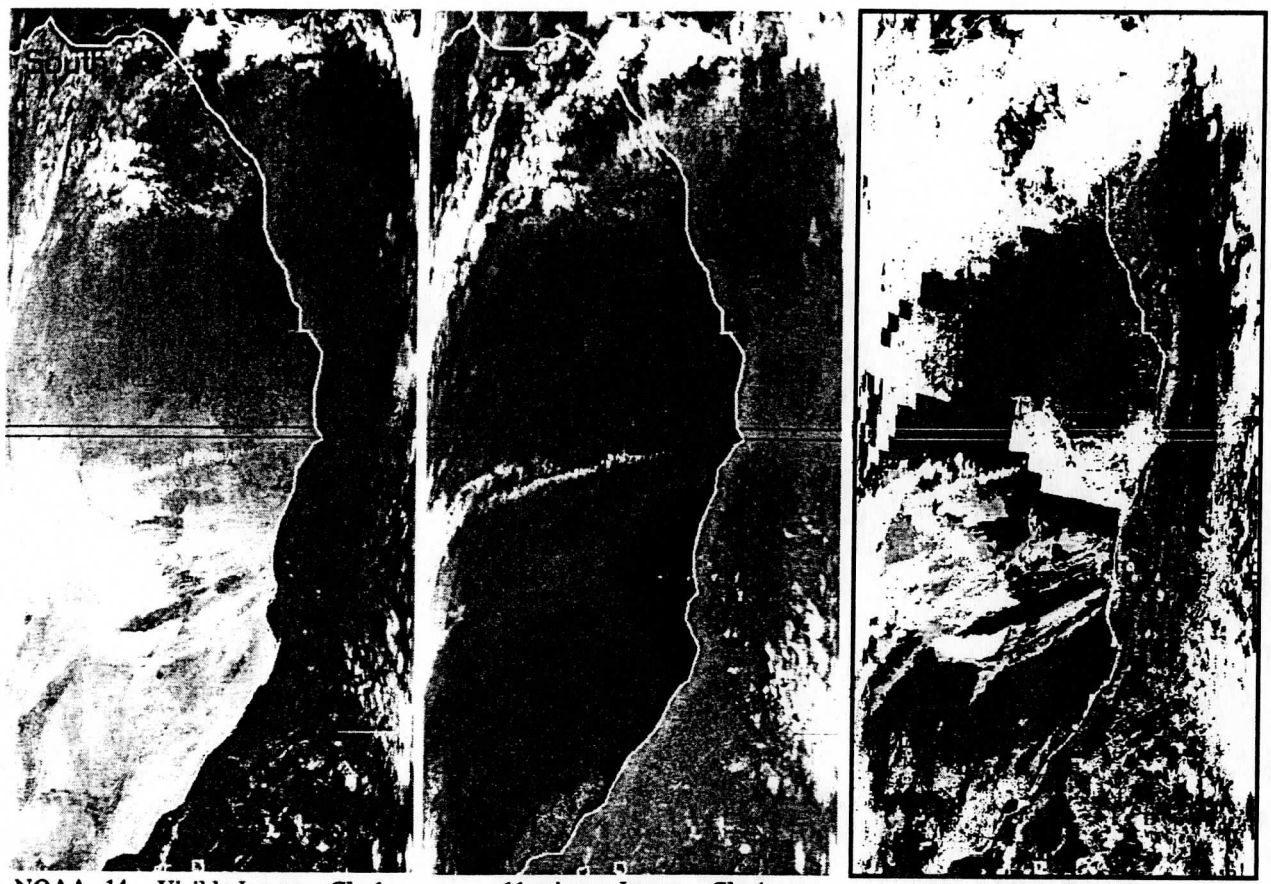

Figure 6. An example cloud mask, together with images at two AVHRR bands, obtained over the eastern Atlantic and west Africa. South is at the top of the figure.

MAS cloud mask data sets

The cloud masking algorithm has also been applied to several MAS scenes. Here we present results from two scenes of the MAS 50 channel data set. The first scene is from observations made on 7 June 1995 over the north slope of Alaska (tundra). The MAS has a spatial resolution of approximate 50 m. Figure 7 includes the visible, infrared, and the final cloud mask. Results from individual tests are available from the authors.

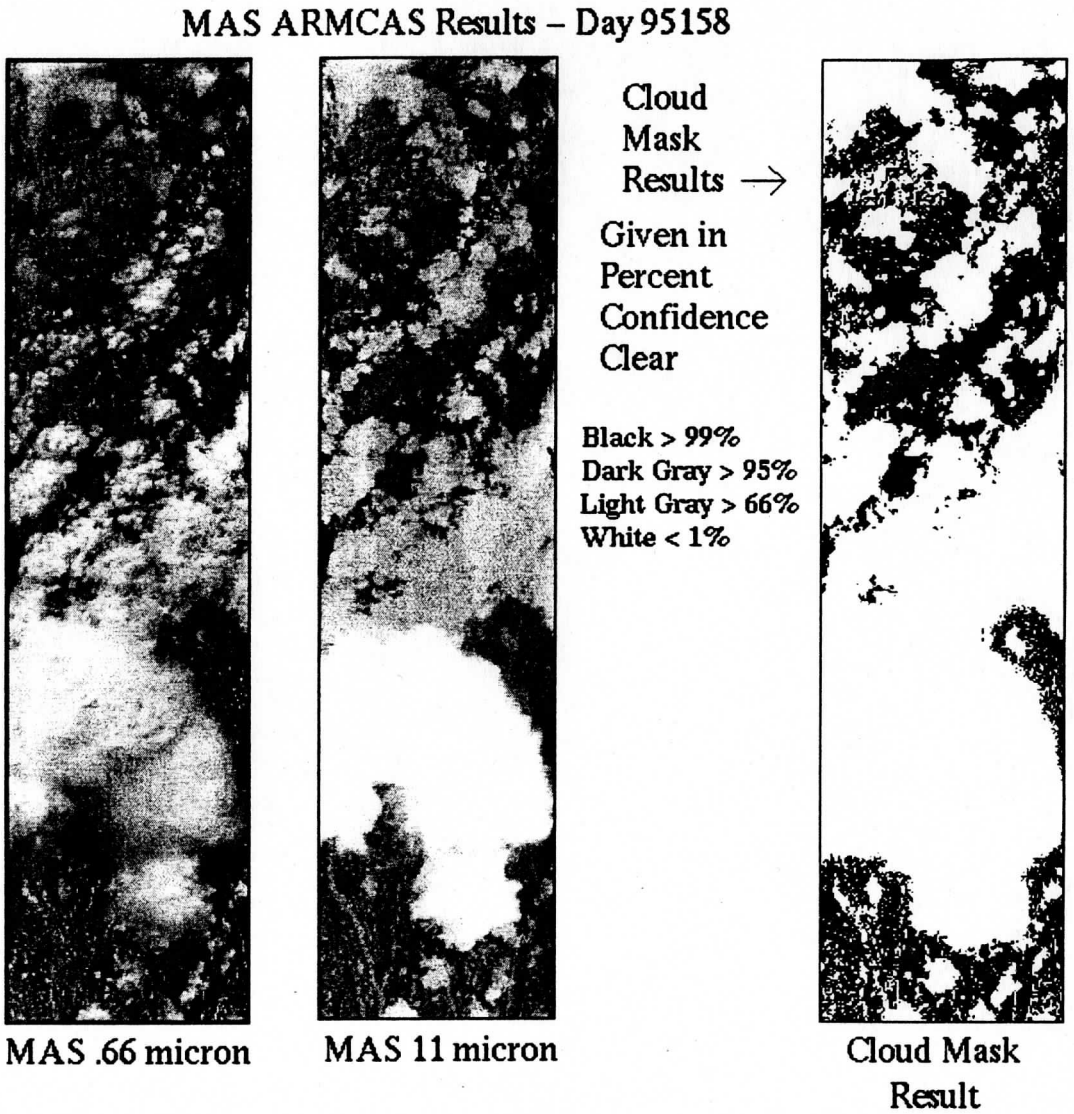

Figure 7. An example cloud mask output using MAS data obtained over the north slope of Alaska on 7 June 1995.

The second example of an application of the MAS cloud mask (Figure 8) is of a scene from ARMCAS provided by Dr. S. -C. Tsay. This scene is cloud free, but includes both dark tundra and light sea ice, thus all pixels indicated as high confident cloudy are false detections. Many of these falsely detected scenes result from failing the 1.88 μm test. Causes and solutions for these failures are under investigation.

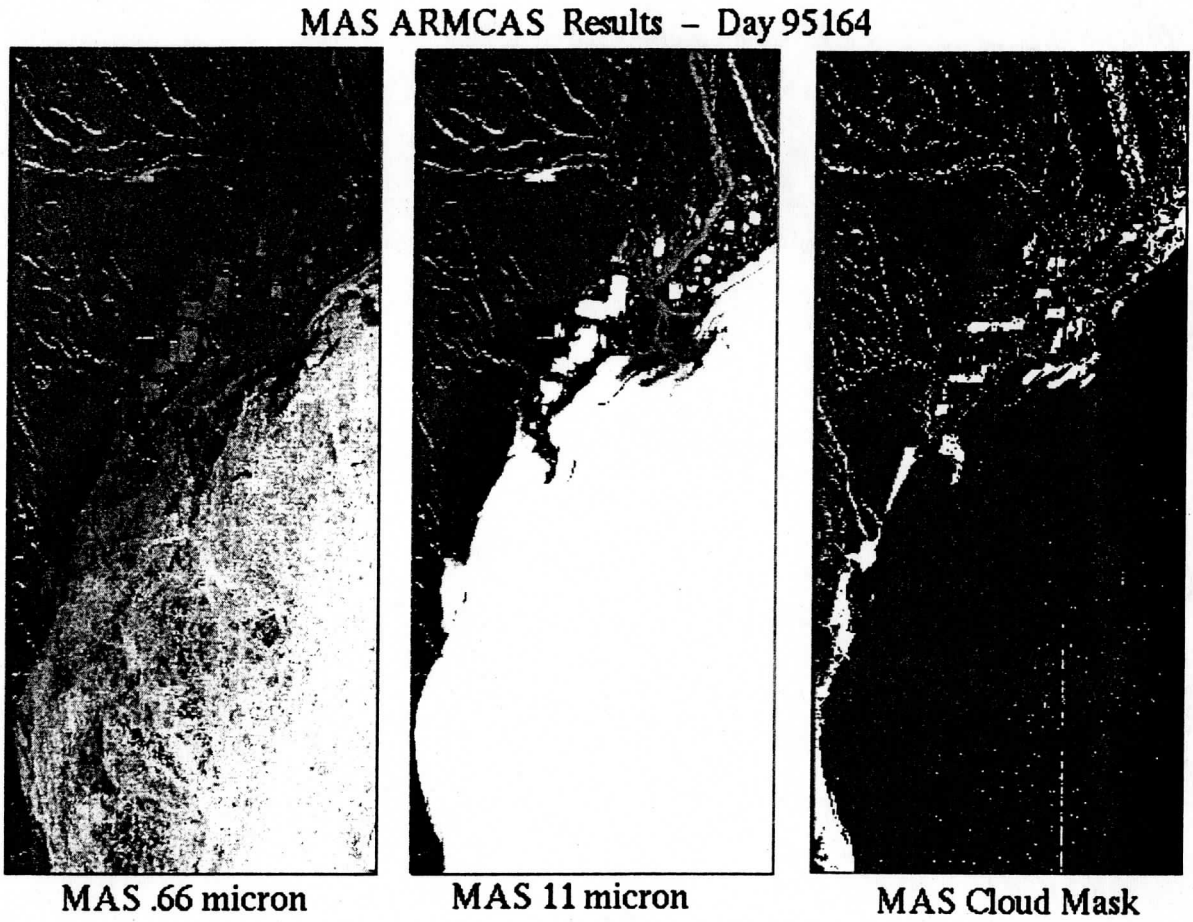

Figure 8. An example cloud mask output using MAS data obtained near Prudhoe Bay and sea ice in the Beaufort Sea on 13 June 1996.

Future MAS Data Processing

The University of Wisconsin-Madison will be applying and validating the MAS cloud mask using data from the recent SUCCESS experiment. The code is available for other GLI team members. The MAS cloud mask code which simulates the GLI cloud mask, and reads the

Goddard MAS HDF data format, will be available to all GLI science team members in August 1997.

4.3 Numerical Programming Considerations

A version of the GLI mask currently operates on MAS data is available for testing. Based on comparison with MODIS cloud mask processing, we estimate the GLI cloud mask to require an output volume load 1.5 GB/day. The product volume is expected to be slightly greater than this with the addition of metadata fields.

4.4 Development Plan

The GLI cloud mask algorithms build on existing programs, particularly the MODIS program; however, much work yet remains to develop and implement the algorithms by launch. Some of the approaches outlined in the document can be developed with current satellite observations, while others require modeling efforts and regional observations with the MODIS Airborne Simulator (MAS).

Our research and development strategy makes use of three data sets. The first is a global data set of collocated HIRS/2 and AVHRR GAC (Global Area Coverage) observations. The strengths of this data set are its global coverage and that the collocated observations have many GLI like bandwidths. This allows us to investigate many of the spectral tests described above. The disadvantage of the data set is the large FOV (HIRS/2 FOV is 17 km at nadir) and the horizontal displacement of the HIRS/2 footprints (gaps of approximately 20 km). This data set also does not have the complete AVHRR image, only those AVHRR pixels that lie within the HIRS/2 footprint. To overcome this later disadvantage we have compiled a second data set of collocated HIRS/2 and AVHRR LAC (Local Area Coverage) observations. This data set has similar spectral and spatial resolution to the future GLI and will be important for validation and development of cloud shadow detection techniques. The disadvantages, of course, are the non-global nature and the gaps between the HIRS FOV.

The third data set is from the MAS aircraft flights. The initial MAS instrument recorded twelve (or eleven) spectral bands in the visible to infrared and enabled development and exploration of some of the techniques discussed above. Since December 1994, the MAS recorded 50 spectral channels, this data will play an important role in testing and developing the mask.

We recognize the need to work closely with the GLI land, ocean, and atmosphere groups in the continued development of the cloud mask. To this end we have offered to process different MAS and AVHRR data scenes for examination. We have developed the FORTRAN code to read the Goddard HDF MAS 50 channel data format and output a cloud mask similar in structure to the GLI 16 bit cloud mask. *This code is available to GLI science team members to test the cloud mask and assist in its development.*

4.5 Validation Plan

Validation will be approached in several ways: (i) collocation with higher resolution aircraft data, (ii) ground-based observations, and (iii) intercomparisons with other AM-1 platform instruments. Our validation approach relies heavily on the sources of data used in the algorithm development, which consisted primarily of the MAS, a fifty channel visible, near-infrared, and thermal infrared imaging spectrometer with 50 m resolution at nadir (King et al. 1996); HIS, a 2 km resolution nadir-viewing Michelson interferometer with 0.5 cm^{-1} spectral resolution from 4 to 15 μm (Revercomb et al. 1988); and AVIRIS, a 224 band imaging spectrometer from 0.4-2.5 μm with 20 m resolution at nadir (Vane et al. 1993). In addition, we plan to make extensive use of the AERONET (Aerosol Robotic Network), a network of ground-based sunphotometers established and maintained by Brent Holben that measure the spectral aerosol optical thickness and sky radiance, reporting the data via a satellite communication link from each remotely-located CIMEL sunphotometer to Goddard Space Flight Center from sunrise to sunset, 7 days a week. Finally, we plan to utilize ground-based radar, lidar, and radiometer observations, especially over the Atmospheric Radiation Measurement (ARM) CART (Clouds And Radiation Testbed) site in Oklahoma.

Comparison of GLI radiances and products with those from other instruments should be made periodically (perhaps annually) in different seasons in daytime and nighttime conditions. We anticipate numerous opportunities, unspecified, in which scientists worldwide will compare

GLI-derived atmospheric, land, and ocean data products with local measurements of the geophysical property of interest. This wide-scale synthesis of data sets obtained from scientists of Australia, Japan, China, Europe, South America, and Africa will greatly enhance the confidence that we place in the GLI-derived products, and will, with time, aid our ability to assess the quality of the data products from a wide variety of climatic conditions and seasons.

5.0 References

Ackerman, S. A., W. L. Smith and H. E. Revercomb, 1990: The 27-28 October 1986 FIRE IFO cirrus case study: Spectral properties of cirrus clouds in the 8-12 micron window. *Mon. Wea. Rev.*, **118**, 2377-2388.

Allen, R. C., Jr., P. A. Durkee and C. H. Wash, 1990: Snow/cloud discrimination with multispectral satellite measurements. *J. Appl. Meteor.*, **29**, 994-1004.

Arking, A., and J. D. Childs, 1985: Retrieval of cloud cover parameters from multispectral satellite images. *J. Climate Appl. Meteor.*, **24**, 322-333.

Baum, B. A., T. Uttal, M. Poellot, T. P. Ackerman, J. M. Alvarez, J. Intrieri, D. O'C. Starr, J. Titlow, V. Tovinkere, and E. Clothiaux, 1995: Satellite remote sensing of multiple cloud layers. *J. Atmos. Sci.*, **52**, 4210-4230.

Baum, B. A., B. A. Wielicki, and P. Minnis, 1992: Cloud-property retrieval using merged HIRS and AVHRR data. *J. Appl. Meteor.*, **31**, 351-369.

Ben-Dor, E., 1994: A precaution regarding cirrus cloud detection from airborne imaging spectrometer data using the 1.38 micron water vapor band. *Remote Sens. Environ.*, **50**, 346-350.

Berendes, T., S. K. Sengupta, R. M. Welch, B. A. Wielicki, and M. Navar, 1992: Cumulus cloud base height estimation from high spatial resolution data: A Hough transform approach. *IEEE Trans. Geosci. Remote Sens.*, **30**, 430-443.

Booth, A. L., 1973: Objective cloud type classification using visual and infrared satellite data. 3rd Conference on Probability and Statistics in the Atmospheric Sciences. Amer. Meteor. Soc., Boulder, CO.

Brooks, D. R., E. F. Harrison, P. Minnis, J. T. Suttles, and R. S. Kandel, 1986: Development of algorithms for understanding the temporal variability of the earth's radiation balance. *Rev. Geophys.*, **24**, 422-438.

Chen, D. W., S. K. Sengupta, and R. M. Welch, 1989: Cloud field classification based upon high spatial resolution textural features, Part 2, Simplified vector approaches, *J. Geophys. Res.*, **94**, 14749-14765.

Coakley, J. A, 1987: A dynamic threshold method for obtaining cloud cover from satellite imagery data. *J. Geophys. Res.*, **92**, 3985-3990.

Coakley, J. A. and F. P. Bretherton, 1982: Cloud cover from high-resolution scanner data: Detecting and allowing for partially filled fields of view. *J. Geophys. Res.*, **87**, 4917-4932.

Derrien, M., B. Fraki, L. Harang, H. Legleau, A. Noyalet, D. Pochic, and A. Sairouni, 1993: Automatic cloud detection applied to NOAA-11 AVHRR imagery. *Remote Sens. Environ.*, **46**, 246-267.

Deschamps, P. Y. and T. Phulpin, 1979: Atmospheric correction of infrared measurements of sea surface temperature using channels at 3.7, 11, and 12 microns. *Boundary Layer Meteor.*, **18**, 131-143.

Ebert, E. E., 1987: A pattern-recognition technique for distinguishing surface and cloud types in the polar-regions. *J. Climate Appl. Meteor.*, **26**, 1412-1427.

Ebert, E., 1989: Analysis of polar clouds from satellite imagery using pattern recognition and a statistical cloud analysis scheme. *J. Appl. Meteor.*, **28**, 382-399.

Franca, G. B., and A. P. Cracknell, 1995: A simple cloud masking approach using NOAA AVHRR daytime data for tropical areas. *Int. J. Remote Sens.*, **16**, 1697-1705.

Francis, Jennifer A., 1994: Improvements to TOVS retrievals over sea ice and applications to estimating Arctic energy fluxes. *J. Geophys. Res.*, **99**, 10395-10408.

Gao, B.-C., A. F. H. Goetz, and W. J. Wiscombe, 1993: Cirrus cloud detection from airborne imaging spectrometer data using the 1.38 micron water vapor band. *Geophys. Res. Lett.*, **20**, 301-304.

Gao, B.-C., and A. F. H. Goetz, 1991: Cloud area determination from AVIRIS data using water vapor channels near 1 micron. *J. Geophys. Res.*, **96**, 2857-2864.

Garand, L., 1988: Automated recognition of oceanic cloud patterns. Part I: Methodology and application to cloud climatology. *J. Climate*, **1**, 20-39.

Giarratano, J. and G. Riley, 1989: Expert systems: Principles and programming, PWS-KENT Publishing Company, Boston, 632pp.

Gustafson, Gary B., Ronald G. Isaacs, Robert P. d'Entremont, Jeanne M. Sparrow, Thomas M. Hamill, Christopher Grassotti, Douglas W. Johnson, Charles P. Sarkisian, Daniel C. Peduzzi, Brian T. Pearson, Vida D. Jakabhazy, James S. Belfiore, Anthony S. Lisa, 1994: Support of environmental requirements for cloud analysis and archive (SERCAA): Algorithm descriptions. Scientific Report No. 2, 28 March 1994. Phillips Laboratory, Directorate of Geophysics, Air Force Material Command, Hanscom Air Force Base, MA.

Gutman, G., D. Tarpley, and G. Ohring, 1987: Cloud screening for determination of land surface characterisitics in a reduced resolution satellite data set. *Int. J. Remote Sens.*, **8**, 859-870.

Hall, D.K., Riggs, G.A. and Salomonson, V.V. 1995: Development of methods for mapping global snow cover using Moderate Resolution Imaging Spectroradiometer data, *Remote Sens. Env.*, **54**,127-140.

Haralick, R. M., K. S. Shammugam, and I. Dinstein, 1973: Textural features for image classification. *IEEE Trans. On Systems, Man, and Cybernetics*, Vol. **SMC-3**, No. 6, 610-621.

Hayden, C. M., G. S. Wade, and T. J. Schmit, 1996: Derived product imagery from GOES-8. *J. Appl. Meteor.*, **35**, 153-162.

Hecht-Nielsen, R., 1990: Neurocomputing. Addison-Wesley, Reading, MA, 430 pp.

Hou, Y-T., K. A. Campana, K. E. Mitchell, S. K. Lang, and L. L. Stowe, 1993: Comparison of an experimental NOAA AVHRR cloud dataset with other observed and forecast cloud datasets. *J. Atmos. Oceanic Technol.*, **10**, 833-848.

Hunt, G. E., 1973: Radiative properties of terrestrial clouds at visible and infrared thermal window wavelengths. *Quart. J. Roy. Meteor. Soc.*, **99**, 346-369.

Hutchison, K. D., and K. R. Hardy, 1995: Threshold functions for automated cloud analyses of global meteorological satelitte imagery. *Int. J. Remote Sens.*, **16**, 3665-3680.

Inoue, T., 1986: On the temperature and effective emissivity determination of semi-transparent clouds by bi-spectral measurements in the 10 micron window region. *J. Meteor. Soc. Japan*, **63**, 88-99.

Inoue, T., 1987: A cloud type classification with NOAA 7 split window measurements. *J. Geophys. Res.*, **92**, 3991-4000.

Inoue, T., 1989: Features of clouds over the Tropical Pacific during the Northern Hemispheric winter derived from split window measurements. *J. Meteor. Soc. Japan*, **67**, 621-637.

Jacobowitz, H. J., 1970: Emission scattering and absorption of radiation in cirrus clouds. Ph.D. thesis, Massachusetts Institute of Technology, 181 pp.

Kaufman, Y. J., 1987: Satellite sensing of aerosol absorption. *J. Geophys. Res.*, **92**, 4307-4317.

Kaufman, Y. J., and C. Sendra, 1988: Algorithm for atmospheric corrections of visible and near IR satellite imagery. *Int. J. Remote Sens.*, **9**, 1357-1381.

Kaufman, Y. J., and T. Nakajima, 1993: Effect of Amazon smoke on cloud microphysics and albedo - analysis from satellite imagery. *J. Appl. Meteo.*, **32**, 729-744.

Key, J. and R. G. Barry, 1989: Cloud cover analysis with Arctic AVHRR data. 1. Cloud detection. *J. Geophys. Res.*, **94**, 8521-8535.

King, M. D., Y. J. Kaufman, W. P. Menzel and D. Tanré, 1992: Remote sensing of cloud, aerosol, and water vapor properties from the Moderate Resolution Imaging Spectrometer (MODIS). *IEEE Trans. Geosci. Remote Sens.*, **30**, 2–27.

King, M. D., W. P. Menzel, P. S. Grant, J. S. Myers, G. T. Arnold, S. E. Platnick, L. E. Gumley, S. C. Tsay, C. C. Moeller, M. Fitzgerald, K. S. Brown and F. G. Osterwisch, 1996: Airborne scanning spectrometer for remote sensing of cloud, aerosol, water vapor and surface properties. *J. Atmos. Oceanic Technol.*, **13**, 777–794.

King, Michael D., Lawrence F. Radke, and Peter V. Hobbs, 1993: Optical properties of marine stratocumulus couds modified by ships. *J. Geophys. Res.*, **98**, 2729-2739.

Kriebel, K. T., 1978: Measured spectral bidirectional reflection properties of four vegetated surfaces. *Appl. Opt.*, **17**, 253-259.

Lee, J., R. Weger, S. K. Sengupta, and R. M. Welch, 1990: A neural network approach to cloud classification. *IEEE Trans. Geosci. and Remote Sens.*, **28**, 846-855.

Lindsay, R. W. and D. A. Rothrock, 1994: Arctic sea ice surface temperature from AVHRR. *J. Climate*, **7**, 174-183.

Liou, K. N., 1973: A numerical experiment on Chandrasekhar's discrete-ordinate method for radiative transfer: Applications to cloudy and hazy atmospheres. *J. Atmos. Sci.*, **30**, 1303-1326.

Liou, K. N., S. C. Ou, Y. Takano, F. P. J. Valero, and T. P. Ackerman, 1990: Remote sounding of the tropical cirrus cloud temperature and optical depth using 6.5 and 10.5 μm Radiometers during STEP. *J. Appl. Meteor.*, **29**, 716-726.

Luger, G. F. and W. A. Stubblefield, 1989: Artificial intelligence and the design of expert systems, The Benjamin Cummings Publishing Company, Inc., 600 pp.

Matthews, E. and W. B. Rossow, 1987: Regional and seasonal variations of surface reflectance from satellite observations at 0.6 mm. *J. Climate Appl. Meteor.*, **26**, 170-202.

McClain, E. P., 1993: Evaluation of CLAVR Phase-I algorithm performance: Final Report, U. S. Department of Commerce/NOAA/NESDIS, Report 40-AANE-201-424.

McMillin, L. M., and C. Dean, 1982: Evaluation of a new operational technique for porducing clear radainces. *J. Appl. Meteor.*, **12**, 1005-1014.

McMillin, Larry, Si-Song Zhou, and Shi-Keng Yang, 1994: An improved cloud retrieval algorithm using HIRS2-MSU radiance measurements. *J. Appl. Meteor.*, **33**, 195-211.

Meerkoetter, R., 1993: Detection of polar stratospheric clouds with next generation IR sounders. NATO ASI Series, 19, 205-213.

Menzel, W. P. and K. I. Strabala, 1989: Preliminary report on the demonstration of the VAS CO_2 cloud parameters (cover, height, and amount) in support of the Automated Surface Observing System (ASOS). NOAA Tech Memo NESDIS 29.

Menzel, W. P., D. P. Wylie, and K. I. Strabala, 1992: Seasonal and diurnal changes in cirrus clouds as seen in four years of observations with the VAS. *J. Appl. Meteor.*, **31**, 370-385.

Menzel, W. P., D. P. Wylie, and K. I. Strabala, 1993: Trends in global cirrus inferred from four years of HIRS data. Technical Proceedings of the Seventh International TOVS Study Conference held 10-16 February in Igls, Austria.

Minnis, P. and E. F. Harrison, 1984a: Diurnal variability of regional cloud and clear sky radiative parameters derived from GOES data. Part I: Analysis method. *J. Climate Appl. Meteor.*, **23**, 993-1011.

Minnis, P. and E. F. Harrison, 1984b: Diurnal variability of regional cloud and clear sky radiative parameters derived from GOES data. Part II: November 1978 cloud distributions. *J. Climate Appl. Meteor.*, **23**, 1012-1031.

Minnis, P., and E. F. Harrison, 1984c: Diurnal variability of regional cloud and clear-sky radiative parameters derived from GOES data, Part III: November 1978 radiative parameters. *J. Climate Appl. Meteor.*, **23**, 1032-1051.

Minnis, P., E. F. Harrison and G. G. Gibson, 1987: Cloud cover over the eastern equatorial Pacific derived from July 1983 ISCCP data using a hybrid bispectral threshold method. *J. Geophys. Res.*, **92**, 4051-4073.

Minnis, P., W. L. Smith, Jr., D. P. Garber, J. K. Ayers, and D. R. Doelling, 1995: Cloud Properties Derived From GOES-7 for Spring 1994 ARM Intensive Observing Period Using Version 1.0.0 of ARM Satellite Data Analysis Program. NASA Reference Publication 1366.

Mokhov, I. I. and M. E. Schlesinger, 1994: Analysis of global cloudiness. 2. Comparison of ground-based and satellite-based cloud climatologies. *J. Geophy. Res.*, **99**, 17045-17065.

Molnar, G. and J. A. Coakley, Jr., 1985: Retrieval of cloud cover from satellite imagery data: a statistical approach. *J. Geophy. Res.*, **90**, 12960-12970.

O'Brien, D. M., and R. M. Mitchell, 1992: Error estimates for retrieval of cloud-top pressure using absorption in the A band of oxygen. *J. Appl. Meteor.*, **31**,1179-1192.

Ou, S. C., K. N. Liou, W. M. Gooch, and Y. Takano, 1993: Remote sensing of cirrus cloud parameters using advanced very-high-resolution radiometer 3.7- and 10.9-um channels. *Appl. Opt.*, **32**, No. 12, 2171-2180.

Parol, F. J., C. Buriez, G. Brogniez and Y. Fouquart, 1991: Information content of AVHRR Channels 4 and 5 with respect to the effective radius of cirrus cloud particles. *J. Appl. Meteor.*, **30**, 973-984.

Phulpin, T., M. Derrien, and A. Baird, 1983: A two-dimensional histogram procedure to analyze cloud cover from NOAA satellite high resolution imagery. *J. Climate Appl. Meteor.*, **22**, 1332-1345.

Prabhakara, C., R. S. Fraser, G. Dalu, M. C. Wu, and R. J. Curran, 1988: Thin cirrus clouds: seasonal distribution over oceans deduced from Nimbus-4 IRIS. *J. Appl. Meteor.*, **27**, 379-399.

Prabhakara, C., J.-M. Yoo, D. P. Kratz, and G. Dalu, 1993: Boundary layer stratus clouds: inferred from satellite infrared spectral measurements over oceans. *J. Quant. Spectrosc. Radiat. Trans.*, **49**, 599-607.

Rao, C. R. N., L. L. Stowe, and L. L. McClain, 1989: Remote sensing of aerosol over the oceans using AVHRR data: theory, practice, and applications. *Int. J. Remote Sens.*, **10**, 743-749.

Rao, N. X., S. C. Ou, and K. N. Liou, 1995: Removal of the solar component in AVHRR 3.7-μm radiances for the retrieval of cirrus cloud parameters. *J. Appl. Meteor.*, **34**, 482-499.

Raschke, E., P. Bauer, and H. J. Lutz, 1992: Remote sensing of clouds and surface radiation budget over polar regions. *Int J. Remote Sens.*, **13**, 13-22.

Reutter, H. , F. -S. Olesen, and H. Fischer, 1994: Distribution of the brightness temperature of land surfaces determined from AVHRR data. *Int. J. Remote Sens.*, **15**, 95-104.

Revercomb, H. E., H. Buijs, H. B. Howell, D. D. LaPorte, W. L. Smith and L. A. Sromovsky, 1988: Radiometric calibration of IR Fourier transform spectrometers: Solution to a problem with the High-spectral resolution Interferometer Sounder. *Appl. Opt.*, **27**, 3210–3218.

Rossow, W. B., F. Mosher, E. Kinsella, A. Arking, M. Desbois, E. Harrison, P. Minnis, E. Ruprecht, G. Seze, C. Simmer, and E. Smith, 1985: ISCCP cloud algorithm intercomparison. *J. Climate Appl. Meteor.*, **24**, 877-903.

Rossow, W. B., 1989: Measuring cloud properties from space. A review. *J. Climate*, **2**, 201-213.

Rossow, W. B., and A. A. Lacis, 1990: Global and seasonal cloud variations from satellite radiance measurements. Part II: Cloud properties and radiative effects. *J. Clim.*

Rossow, W. B. and L. C. Garder, 1993: Cloud detection using satellite measurements of infrared and visible radiances for ISCCP. *J. Climate*, **6**, 2341-2369.

Rossow, William B. and Leonid. C. Garder, 1993: Validation of ISCCP cloud detections. *J. Climate*, **6**, 2370-2393.

Rossow, William B., Alison W. Walker, and Leonid C. Garder, 1993: Comparison of ISCCP and other cloud amounts. *J. Climate*, **6**, 2394-2418.

Sakellariou, N. K., H. G. Leighton, and Z. Li, 1993: Identification of clear and cloudy pixels at high latitudes from AVHRR radiances. *Int. J. Remote Sens.*, **14**, 2005-2024.

Saunders, R. W., 1986: An automated scheme for the removal of cloud contamination from AVHRR radiances over western Europe. *Int. J. Remote Sens.*, **7**, 867-886.

Saunders, R. W. and K. T. Kriebel, 1988: An improved method for detecting clear sky and cloudy radiances from AVHRR data. *Int. J. Remote Sens.*, **9**, 123-150.

Schmetz, J., C. Geijo, W.P. Menzel, K. Strabala, L. Van De Berg, K. Holmlund, and S. Tjemkes, 1995: Satellite observations of upper tropospheric relative humidity, clouds and wind field divergence. *Beitr. Phys. Atmos.*, Vol. **68**, No. 4, 345-357.

Seze, G. and M. Desbois, 1987: Cloud cover analysis from satellite imagery using spatial and temporal characteristics of the data. *J. Climate Appl. Meteor.*, **26**, 287-303.

Seze, G., and W. B. Rossow, 1991a: Time-cumulated visible and infrared radiance histograms used as descriptors of surface and cloud variations. *Int. J. Remote Sens.*, **12**, 877-920.

Seze, G., and W. B. Rossow, 1991b: Effects of satellite data resolution on measuring the space-time variations of surfaces and clouds. *Int. J. Remote Sens.*, **12**, 921-952.

Simpson, J. J. and J. I. Gobat, 1995: Improved cloud detection in GOES scenes over land. *Remote Sens. Environ.*, **52**, 36-54.

Simpson, J. J., and C. Humphrey, 1990: An automated cloud screening algorithm for daytime advanced very high resolution radiometer imagery. *J. Geophys. Res.*, **95**, 13459-13481.

Smith, W. L. and C. M. R. Platt, 1978: Comparison of satellite-deduced cloud heights with indications from radiosonde and ground-based laser measurements. *J. Appl. Meteor.*, **17**, 1796-1802.

Smith, W. L, 1968: An improved method for calculating tropospheric temperature and moisture profiles from satellite radiometer measurements. *Mon. Wea. Rev.*, **96**, 387.

Spinhirne, J. D., R. Boers and W. D. Hart, 1989: Cloud top liquid water from lidar observations of marine stratocumulus. *J. Appl. Meteor.*, **28**, 81–90.

Stephens, Graeme L., 1980: Radiative properties of cirrus clouds in the infrared region. *J. Atmos. Sci.*, **37**, 435-446.

Stephens, G. L., 1990: On the relationship between water vapor over the oceans and sea surface temperature. *J. Climate*, **3**, 634-645.

Stone, Robert S., Graeme L. Stephens, C.M.R. Platt, and S. Banks, 1990: The remote sensing of thin cirrus cloud using satellites, lidar and radiative transfer theory. *J. Appl. Meteor.*, **29**, No. 5, 353-366.

Stowe, L. L., H. Y. M. Yeh, C. G. Wellemeyer, H. L. Kyle, and the Nimbus-7 Cloud Data Processing Team, 1989: Nimbus-7 global cloud climatology, Part II: First year results. *J. Climate*, **2**, 671-709.

Stowe, L. L., E. P. McClain, R. Carey, P. Pellegrino, G. Gutman, P. Davis, C. Long, and S. Hart, 1991: Global distribution of cloud cover derived from NOAA/AVHRR operational satellite data. *Adv. Space Res.*, **11**, 51-54.

Stowe, L. L., R. M. Carey, and P. Pellegrino, 1992: Monitoring the Mt. Pinatubo aerosol layer with NOAA/11 AVHRR data. *Geophys. Res. Lett.*, **19**, 159-162

Stowe, L. L., S. K. Vemury, and A. V. Rao, 1994: AVHRR clear sky radiation data sets at NOAA/NESDIS. *Adv. Space Res.*, **14**, 113-116.

Stowe, L. L., P. Davis, and E. P. McClain, 1995: Evaluating the CLAVR (Clouds from AVHRR) Phase I cloud cover experimental product. *Adv. in Space Res.*, **16**, 21-24.

Strabala, K. I., S. A. Ackerman, and W. P. Menzel, 1994: Cloud properties inferred from 8-12 μm data. *J. Appl. Meteor.*, **33**, 212-229.

Sun, C., and Wee, W.G., 1983: Neighboring grey level dependence matrix for texture classification. *Comp. Vision, Graphics, Image Proc.*, **23**, 341-352.

Susskind, J., D. Reuter, and M. T. Chahine, 1987: Cloud fields retrieved from analysis of HIRS/MSU sounding data. *J. Geophys. Res.*, **92**, 4035-4050.

Suttles, J. T., R. N. Green, P. Minnis, G. L. Smith, W. F. Staylor, B.A. Wielicki, I. J. Walker, D. F. Young, V. R. Taylor, and L. L. Stowe, 1988: Angular radiation models for Earth-atmosphere system: Volume I - Shortwave radiation. NASA RP 1184, 144 pp.

Takano, Y., K. N. Liou and P. Minnis, 1992: The effects of small ice crystals on cirrus infrared radiative properties. *J. Atmos. Sci.*, **49**, 1487-1493.

Tanré, D., P. Y. Deschanps, C. Devaux, and M. Herman, 1988: Estimation of Saharan aerosol optical thicknesses from blurring effects in Thematic Mapper data. *J. Geophys. Res.*, **93**, 15955-15964.

Tarpley, J. D., 1979: Estimating incident solar radiation at the surface from geostationary satellite data. *J. Appl. Meteor.*, **18**, 1172-1181.

Tjemkes, Stephen A., Leo van de Berg, and Johannes Schmetz, 1996: Simultaneous observations of cold clouds in METEOSAT water vapour and infrared window channel. Submitted to *Contr. to Atmos. Phys.* for publication as a note.

Tovinkere, V. R., M. Penaloza, A. Logar, J. Lee, R. C. Weger, T. A. Berendes, and R. M. Welch, 1993: An intercomparison of artificial intelligence approaches for polar scene identification, *J. Geophys. Res.*, **98**, 5001-5016.

Udelhofen, Petra M. and Dennis L. Hartmann, 1995: Influence of tropical cloud systems on the relative humidity in the upper troposphere. *J. Geophys. Res.*, **100**, 7423-7440.

Vane, G., R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, and W. M. Porter, 1993: The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS). *Remote Sens. Environ.*, **44**, 127–143.

Vidal, A., 1991: Atmospheric and emissivity correction of land surface temperature measured from satellite using ground measurements or satellite data. *Int. J. Remote Sens.*, **12**, 2449-2460.

Wang, J., and W.B. Rossow, 1995: Determination of cloud vertical structure from upper-air observations. *J. Appl. Meteor.*, **34**, 2243-2258.

Warren, S. G., 1984: Optical constants of ice from the ultraviolet to the microwave. *Appl. Opt.*, **23**, 1206-1225.

Welch, R. M., S. K. Sengupta, A. K. Goroch, P. Rabindra, N. Rangaraj, and M. S. Navar, 1992: Polar cloud and surface classification using AVHRR imagery: An intercomparison of methods, *J. Appl. Meteor.*, **31**, 405-420.

Welch, R. M., K. S. Kuo, S. L. Sengupta, 1990: Cloud and surface textural features in polar regions. *IEEE Trans. Geosci., and Remote Sens.*, **28**, 520-528.

Welch, R. M., M. S. Navar, and S. K. Sengupta, 1989: The effect of spatial resolution upon texture-based cloud field classifications. *J. Geophys. Res.*, **94**, 14767-14781.

Welch, R.M., et al., 1988: Cloud field classification based upon high spatial resolution textural features, Part 1, Gray level cooccurrence matrix approach. *J. Geophys. Res.*, **93**, 12663-12681.

Wielicki, B. A., and J. A. Coakley, 1981: Cloud retrieval using infrared sounder data: Error analysis. *J. Appl. Meteor.*, **20**, 157-169.

Wylie, D. P., and W. P. Menzel, 1989: Two years of cloud cover statistics using VAS. *J. Climate Appl. Meteor.*, **2**, 380-392.

Wylie, D. P., W. P. Menzel, H. M. Woolf, and K. I. Strabala, 1994: Four years of global cirrus cloud statistics using HIRS. *J. Climate*, **7**, 1972-1986.

Yamanouchi, T., K. Suzuki, and S. Kawaguci, 1987: Detection of clouds in Antarctica from infrared multispectral data of AVHRR. *J. Meteor. Soc. Japan*, **65**, 949-962.

Appendix A. Acronyms

ACARS	ARINC (Aeronautical Radio Inc.) Communications, Addressing and Reporting System
AERI	Atmospheric Emitted Radiation Interferometer
AEROCE	Aerosol/Ocean Chemistry Experiment
AERONET	Aerosol Robotic Network
AirMISR	Airborne MISR
AIRS	Atmospheric Infrared Sounder
AMSU	Advanced Microwave Sounding Unit
APOLLO	AVHRR (Advanced Very High Resolution Radiometer) Processing scheme Over cLoud Land and Ocean
ARM	Atmospheric Radiation Measurement Program
ARMCAS	Arctic Radiation Measurements in Column Atmosphere-surface System (Beaufort Sea, Alaska, June 1995)
ASTEX	Atlantic Stratocumulus Transition Experiment (Azores, June 1992)
ASTER	Advanced Spaceborne Thermal Emission and Reflection radiometer
AVHRR	Advanced Very High Resolution Radiometer
AVIRIS	Airborne Visible/Infrared Imaging Spectrometer
CAR	Cloud Absorption Radiometer
CART	Clouds and Radiation Testbed
CEPEX	Central Equatorial Pacific Experiment (Fiji, February-March 1993)
CERES	Clouds and the Earth's Radiant Energy System
CHAPS	Collocated HIRS/2 and AVHRR Processing Scheme
CLAVR	Cloud Advanced-Very High Resolution Radiometer
CLS	Cloud Lidar System
COARE	Coupled Ocean-Atmosphere Response Experiment
DAO	Data Assimilation Office (Goddard Space Flight Center)
EOS	Earth Observing System
EOSDIS	EOS Data and Information System

FIRE	First ISCCP Regional Experiment (California, June-July 1987, Beaufort Sea, Alaska, April-June, August 1998)
FOV	Field of View
GAC	Global Area Coverage
GLAS	Geoscience Laser Altimeter System
GLI	Global Imager
GOES	Geostationary Operational Environmental Satellite
HIS	High-spectral resolution Interferometer Sounder
HIRS	High Resolution Infrared Radiation Sounder
HSB	Humidity Sounder from Brazil
ILAS	Improved Limb Atmospheric Spectrometer
ISCCP	International Satellite Cloud Climatology Project
LASE	Lidar Atmospheric Sensing Experiment
LBA	Large Scale Biosphere-Atmosphere Experiment in Amazonia
MAS	MODIS Airborne Simulator
MAST	Monterey Area Ship Tracks Experiment (Monterey and nearby Pacific Ocean, June 1994)
McIDAS	Man-computer Interactive Data Access System
MISR	Multi-angle Imaging Spectro-Radiometer
MOBY	Marine Optical Buoy
MODIS	Moderate Resolution Imaging Spectroradiometer
NAST	NPOESS Aircraft Sounding Testbed
NCAR	National Center for Atmospheric Research
NDSI	Normalized Difference Snow Index
NDVI	Normalized Difference Vegetation Index
NPOESS	National Polar Orbiting Environmental Satellite System
POLDER	Polarization and Directionality of Earth's Reflectances
RAMS	Radiation Measurement System (NASA Ames Research Center and Scripps Institution of Oceanography)

SCAR-A	Sulfate, Clouds and Radiation–Atlantic (Delmarva Peninsula and near-by Atlantic Ocean, July 1993)
SCAR-B	Smoke, Clouds and Radiation–Brazil (Brazil, August-September 1995)
SCAR-C	Smoke, Clouds and Radiation–California (Pacific Northwest, September 1994)
SCF	Science Computing Facility
SeaWiFS	Sea-viewing Wide Field-of-view Sensor
SHEBA	Surface Heat Budget of the Arctic Ocean
SST	Sea Surface Temperature
SUCCESS	Subsonic Aircraft Contrail and Cloud Effects Special Study (April-May 1996)
TARFOX	Tropospheric Aerosol Radiative Forcing Observational Experiment (Delmarva Peninsula and near-by Atlantic Ocean, July 1996)
TIROS	Television and Infrared Observation Satellite
TLCF	Team Leader Computing Facility
TM	Thematic Mapper
TOGA	Tropical Ocean Global Atmosphere
TOMS	Total Ozone Mapping Spectrometer
TOVS	TIROS-N Operational Vertical Sounder
WINCE	Winter Cloud Experiment
WMO	World Meteorological Organization

LISTING OF VERSION 1 OF THE GLI CLOUD MASK

```
################################################################################
#
#       README for ATSK1.exe
#
#       Oct.22,1998  Fujitsu Ltd.
#
################################################################################

The original main module "masglimsk.f" is replaced by "ATSK1.f".
"ATSK1.f" is consists of 5 submodules and 1 include file.

[New Added Files]
cldmask.inc
        - Define parameters.

ATSK1.f
        - Main module.
        - Get run parameter(Command line option).

init_proc.f
        - Open input/output files.
        - Initialize variables
        - Read supplement data in a image data file.

read_proc.f
        - Read image data every N lines.
        - Read land/sea tag and ecosystem type for necessary
        pixels on the current scan line.

mainlp_proc.f
        - Calculate cloud flag data every N lines.

write_proc.f
        - Write beginning date/time of data to header area
        in the output file.
        - Write cloud flag data to data area in the output file every N line.

stop_proc.f
        - Write number of total lines of cloud flag data to output file.
        - Close every files.

atsk1.fj.mk
        - Makefile for "ATSK1.exe".
        - Macro "HDFHOME" is set to "/opt/local/HDF".


[Instruction]

Modified makefile to set macro "HDFHOME".

> vi atsk1.fj.mk
> make -f atsk1.fj.mk

You get a load module "ATSK1.exe". Execute it.

> ATSK1.exe 970209_08.hdf 970209_08.cm 2 790 2 714 1 0.0

"ATSK1.exe" has just the same Run Parameter as "masglimsk.exe".
You obtain cloud flag data in "970209_08.cm".
```

```
      program ATSK1

c     A program which creates a cloud mask from MAS spectral data. Output
c     is a series of 16 "bit" flags for each MAS FOV. The flags are arranged
c     into 1432-byte records and written to a binary direct access file, one
c     record for each scan line. Some ancillary data is included in the first
c     record.

c     Input data is stored in HDF-format files.

c     Input ancillary filenames are defined in the "file_open.f" subroutine.

c     This version of the MAS cloud mask processes the data pixel by pixel,
c     but in the context of surrounding observations. The size of the "context",
c     or region, is given by the variables 'nlcntx' (# lines in context) and
c     'necntx' (# pixels in context). The sizes must be odd integer numbers and
c     may not both be set to 1.

c**********************************************************************

c     Command line positional parameters:

c         input HDF file name
c         output file name
c         beginning scan line #
c         total # scan lines to process
c         beginning pixel #
c         total # pixels to process
c         ecosystem file index (1=NA 1km, otherwise global 10 minute)
c         pw value (g/cm-2) for tri-spectral cloud test
c            (optional - input value < 1.0 to disable)
c            ** must enter real number **
c
c**********************************************************************

c      MAS Bands used: visible
c            Mas         Array         Central
c         Channel#       index #     Wavelength (um)
c            1             1             .56
c            2             2             .66
c            7             3             .88
c            9             4             .945
c            10            5            1.62
c            15            6            1.88
c            20            7            2.14
c
c      MAS Bands used: infrared
c            Mas         Array         Central
c         Channel#       index #     Wavelength (um)
c            31            8            3.75
c            32            9            3.90
c            42            10           8.54
c            45            11          10.98
c            46            12          11.93
c            49            13          13.66

c    *** CAUTION: channel numbers may change. Use array 'inchan' to vary
c                 channel numbers corresponding to particular MAS spectral
c                 bands. See data statement below.

c**********************************************************************

      include 'cldmask.inc'

c**********************************************************************

      external integer init_proc
      external integer read_proc
      external integer mainlp_proc
      external integer write_proc
      external integer stop_proc

      integer*2 indat(npixel,nbands,linmax)

      integer*4 ibele,nele,if_hdfid,tp_unit,eco_unit,outf_unit,
     *          ibhms,iyrday,jday,maxlin,ibls,ibes,nelin,nelet,
     *          inchns(inband,linmax),islhms1,
     *          lst(npixel,nlcntx,linmax),
     *          pcwatr(npixel,nlcntx,linmax),
     *          procflg,npix,n1sm,n2sm,n3sm,n4sm,outrec(outmax),
     *          start(3),edge(3),gstart(3),gedge(3),
     *          block,count,eco_index

      integer*4 dataid,latid,lonid,szaid,vzaid,solazid,
     *          snsazid,sltid

      real*4 refang(npixel,nlcntx,linmax),cwl(50),scf(50),solirr(50)
      real*4 rlats(npixel,nlcntx,linmax),rlons(npixel,nlcntx,linmax),
     *       sza(npixel,nlcntx,linmax),vza(npixel,nlcntx,linmax),
     *       raz(npixel,nlcntx,linmax),pw

      byte bitarray(2,npixel,linmax),ecstypes(npixel,nlcntx,linmax)

      character*50 parm(8)
      character*50 out_file,in_file
      integer*4 len1,len2,ibl,nlin, iret

c**********************************************************************
c
c     Get input from keyboard.
c
      do num_in = 1,8
        call getarg(num_in,parm(num_in))
      enddo

      read(parm(1),'(a50)') in_file
      read(parm(2),'(a50)') out_file
      read(parm(3),'(i10)') ibl
      read(parm(4),'(i10)') nlin
      read(parm(5),'(i10)') ibele
      read(parm(6),'(i10)') nele
      read(parm(7),'(i10)') eco_index
      read(parm(8),'(f10.5)') pw

      len1 = len(in_file) - 1
      len2 = len(out_file) - 1

c**********************************************************************
c
c   initiate section
c
       iret =       init_proc(ibele,nele,if_hdfid,tp_unit,eco_unit,
     *                        outf_unit,ibhms,iyrday,jday,maxlin,ibls,
     *                        ibes,nelin,nelet,procflg,pw,cwl,scf,solirr,
     *                        start,edge,gstart,gedge,eco_index,dataid,
     *                        latid,lonid,szaid,vzaid,solazid,snsazid,
     *                        sltid,block,
     *                        in_file,len1,out_file,len2,ibl,nlin)

c**********************************************************************
c
c   main-loop section
c
c     write(*,'(1x,''ATSK1 start'')')

C     ibls = 1, nelin = 792
```

```
      do count = 0, block

         iret =    read_proc(tp_unit,eco_unit,maxlin,ibls,nelin,ibes,
     *                       nelet,inchns,islhms1,lst,pcwatr,ecstypes,
     *                       refang,rlats,rlons,sza,vza,raz,indat,
     *                       start,edge,gstart,gedge,eco_index,dataid,
     *                       latid,lonid,szaid,vzaid,solazid,snsazid,
     *                       sltid,block,count)

c         write(*,*) 'read_proc done'

         iret =    mainlp_proc(ibls,nelin,ibes,nelet,inchns,refang,nele,
     *                         maxlin,ibele,pcwatr,ecstypes,lst,jday,
     *                         pw,rlats,rlons,sza,vza,raz,indat,cwl,
     *                         scf,solirr,npix,n1sm,n2sm,n3sm,n4sm,
     *                         bitarray,outrec,block,count)

c         write(*,*) 'mainlp_proc done'

         iret =    write_proc(maxlin,outf_unit,ibls,nelin,bitarray,
     *                        iyrday,ibhms,
     *                        islhms1,outrec,procflg,block,count)

c         write(*,*) 'write_proc done'
      enddo

c*********************************************************************
c
c   stop section
c
      iret =       stop_proc(nelin,outf_unit,bitarray,iyrday,ibhms,
     *                       islhms1,outrec,npix,n1sm,n2sm,n3sm,n4sm,
     *                       if_hdfid,tp_unit,eco_unit,procflg)

c*********************************************************************

      write(*,'(1x,''ATSK1 finished'')')

      stop
      end
```

```
      subroutine DesertDay(nbands,pxldat,vza,visusd,avgtherm,
     +                     testbits,confdnc)

c F77 ********************************************************
c
c Description:
c      Routine for performing clear sky threshold tests over desert
c      surfaces during daylight hours.
c
c Input parameters:
c nbands         Number of MAS channels
c pxldat         Array containing reflectance or brightness temperatures
c                for all bands for a single pixel
c avgtherm       Average BT over region
c vza            Current pixel viewing angle
c visusd         Logical variable indicating whether vis data used or not
c
c Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c End----------------------------------------------------------------

      include 'thresholds.inc'

c ... scalar arguments ..
      integer nbands
      real confdnc,vza
      logical visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masv188,masv66,masv88,
     +     pi,schi,vrat,c6,cmin1,cmin2,cmin3,cmin4,hiconf,loconf,
     +     fac,pre_confdnc,groups

      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit,clear_bit
c ...
c ... intrinsic functions ..
      intrinsic cos,acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv188 = pxldat(6)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent group test confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... Perform tests.
c
C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... Co2 high cloud test.
      if (masir13.gt.dsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dsco2(1),dsco2(3),dsco2(4),
     *               dsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  *************************
c
c
c
c     ****  GROUP 2 TESTS  ***********************************

      nptests = 0

c ... 11-12um brightness temperature difference test
c ... for thin cirrus.
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... Interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... If a threshold was determined, then use this
c ... as the thin cirrus test, otherwise use a standard threshold.
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = ds11_12hi(2)
      else
        dfthrsh = diftemp
      end if

      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
```

```
      end if
      loconf = dfthrsh + 0.5
      hiconf = dfthrsh - 1.25
      call conf_test(masdf1,loconf,hiconf,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir11 .le. 320.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.ds11_41o(2) .or. mas11_4.le. ds11_4hi(2)) then
            nptests = nptests + 1
          end if
          locut(1) = ds11_41o(1)
          locut(2) = ds11_4hi(1)
          hicut(1) = ds11_41o(3)
          hicut(2) = ds11_4hi(3)
          midpt(1) = ds11_41o(2)
          midpt(2) = ds11_4hi(2)

          call conf_test(mas11_4,locut,hicut,1.0,midpt,2,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        endif
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ****************************
c
c
c
c ********  START OF GROUP 3 TESTS ****************************

      nptests = 0

c ... visible (.88 micron) reflectance threshold test.
      if (visusd) then
        if (masv88.le.dsref2(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv88,dsref2(1),dsref2(3),dsref2(4),
     +                 dsref2(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS   ****************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS  ************************

      nptests = 0

c ... near-infrared high cloud test
      if (visusd) then
        if (masv188.le.dsref3(2)) then
          nptests = nptests + 1
        end if
```

```
        call conf_test(masv188,dsref3(1),dsref3(3),dsref3(4),
     +                 dsref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************   END OF GROUP 4 TESTS   ****************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine DesertDay_c(nbands,pxldat,vza,visusd,avgtherm,
     +                       testbits,confdnc)

C!F77 ********************************************************
c
c Description:
c      Routine for performing clear sky tests over coastal desert
c      surfaces during daylight hours.
c
c Input parameters:
c nbands         Number of MAS channels
c pxldat         Array containing reflectance or brightness temperatures
c                for all bands for a single pixel
c avgtherm       Average BT over region
c vza            Current pixel viewing angle
c visusd         Logical variable indicating whether vis data used or not
c
c Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c!End-----------------------------------------------------------------

      include 'thresholds.inc'      ,

c ... scalar arguments ..
      integer nbands
      real confdnc,vza
      logical visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masv188,masv66,masv88,
     +     pi,schi,vrat,c6,cmin1,cmin2,cmin3,cmin4,hiconf,loconf,
     +     fac,pre_confdnc,groups

      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit,clear_bit
c ...
c ... intrinsic functions ..
      intrinsic cos,acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv188 = pxldat(6)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... perform tests.
c
c     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.dsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dsco2(1),dsco2(3),dsco2(4),
     *               dsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c
c     ****  GROUP 2 TESTS  *********************************

      nptests = 0

c ... 11-12um brightness temperature difference test
c ... for thin cirrus
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... if a threshold was determined, then use this
c ... as the thin cirrus test, otherwise use a standard threshold
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = ds11_12hi(2)
      else
        dfthrsh = diftemp
      end if

c...  Set flag if test passed
      if (masdf1.le.dfthrsh) then
```

```
            c = 1.0

         else if(val .lt. alpha1 .or. val .gt. alpha2) then

            c = 0.0

         else if(val .le. gamma1) then

c           Value is within range of lower set of limits.

            if(val .le. beta1) then
               range = 2.0 * (beta1 - alpha1)
               s1 = (val - alpha1) / range
               c = coeff * s1**power
            else
               range = abs(2.0 * (beta1 - gamma1))
               s1 = abs((val - gamma1) / range)
               c = 1.0 - (coeff * s1**power)
            end if

         else

c           Value is within range of upper set of limits.

            if(val .ge. beta2) then
               range = 2.0 * (beta2 - alpha2)
               s1 = (val - alpha2) / range
               c = coeff * s1**power
            else
               range = 2.0 * (beta2 - gamma2)
               s1 = (val - gamma2) / range
               c = 1.0 - (coeff * s1**power)
            end if

         end if

      end if

   else

      write(*,'(1x,''Invalid number of thresholds'')')
      return

   end if

c  Force confidence values to be between 0 and 1.

   if(c .gt. 1.0) c = 1.0
   if(c .lt. 0.0) c = 0.0

   conflev = c

   return
   end
```

```
#-------------------------------------------------
# FORTRAN Makefile for UW MAS Cloud Mask Software
# 02/98
#-------------------------------------------------

# Main program name

MAIN = ATSK1.exe

# Object file names (use 'ls -1x *.f' to generate)

OBJS = \
\
DesertDay.o        DesertDay_c.o     LandDay_c.o       LandDay_type1.o\
LandNite_type1.o   PolarDay_snow.o   PolarNite_snow.o   atherm.o\
bright50.o         chk_cnstnc.o      chk_input.o       clear_bit.o\
coast_day.o        coast_nite.o      conf_test.o       getcoord.o\
dhr2hms.o          file_close.o      file_open.o       get_data.o\
get_geo.o          get_pxldat.o      get_refl3.o       get_regdif.o\
get_sfc.o          get_stats.o       getsctm.o         land_day.o\
land_nite.o        noncld_obs_chk.o  numday.o\
ocean_day.o        ocean_nite.o      polar_nite.o      proc_path.o\
pxl_anc.o          reg_anc.o         reg_data.o        nacoord.o\
rega.o             regb.o            set_HDF.o         set_bit.o\
set_confdnc.o      shadows.o         snowb.o\
solr_irrad.o       spatial_var.o     stats_out.o       tview.o\
water_day.o        water_nite.o      write_bits.o \
init_proc.o        mainlp_proc.o     analysis_proc.o   stop_proc.o\
read_proc.o        write_proc.o\
ATSK1.o

# Compiler options for SGI

FC = f77
#FFLAGS = -n32 -O -check_bounds -bytereclen -lfpe
#FFLAGS = +U77
#FFLAGS = -u
FFLAGS = -O
CC = cc
#CFLAGS = -n32
CFLAGS = -O
# ANSI type C language for HP
#CFLAGS = -Aa

# Toolkit locations at UW

#TOOLDIR = /home4/modis/toolkits
#TOOLDIR = /usr03/space/ksato/hdf
#HDFHOME = $(TOOLDIR)/lib
HDFHOME = /opt/local/HDF

#----------------------------------------
# No edits are necessary after this point
#----------------------------------------

# Include file locations

HDFINC = $(HDFHOME)/include
INCS = -I$(HDFINC)

# Library locations

HDFLIB = $(HDFHOME)/lib
#LIBS = -lm -Wl,-L$(HDFLIB) -lmfhdf -ldf -ljpeg -lz
#LIBS = -lm -L$(HDFLIB) -lmfhdf -ldf -ljpeg -lz
LIBS = -L$(HDFLIB) -lmfhdf -ldf -ljpeg -lz -L/usr/lib -lnsl

# Rules

$(MAIN):        $(OBJS)
        $(FC) -o $@ $(FFLAGS) $(OBJS) $(LIBS)

clean:
        -rm -f *.o $(MAIN)

.SUFFIXES: .o .c .f

.f.o:
        $(FC) -c $(FFLAGS) $(INCS) $<

.c.o:
        $(CC) -c $(CFLAGS) $(INCS) $<
```

```
c     Include file for use in this program.

c     Set # lines and # elements for regional "context". Do not set
c     both 'nlcntx' and 'necntx' to 1.
      parameter(nlcntx=3)
      parameter(necntx=3)

c     Set number of pixels in output cloud flag lines.
      parameter(npixel = 716)

c     Set number of pixels in input MAS lines.
      parameter(in_recl = 716)

c     Set the total number of MAS channels.
      parameter(nbands = 50)

c     Set number of bands used as input.
      parameter(inband = 13)

c     Define midpoint of scan for sub-satellite point calculations.
      parameter(mdele1 = 357)
      parameter(mdele2 = 358)

c     Set number of processing lines.
      parameter(linmax = 12)

c     Set maximum number of processing lines.
      parameter(outmax = 1500)
```

```
c     Include file for use in sun glint conditions (gocndaya.for).

      real*4 snglnt2(36,2),snglnt2cl(36,2),snglnt2ch(36,2),
     *       snglntv(36,2),snglntvcl(36,2),snglntvch(36,2),
     *       snglntd(36,2),snglntdcl(36,2),snglntdch(36,2)

      data snglnt2 /19.5,18.5,30.0,30.0,30.0,30.0,30.0,30.0,
     *              30.0,30.0,25.0,25.0,25.0,25.0,25.0,25.0,
     *              20*10.0,36*0.0/
      data snglnt2cl /25.0,25.0,40.0,60.0,60.0,60.0,50.0,50.0,
     *                50.0,50.0,6*35.0,20*13.5,36*0.0/
      data snglnt2ch /14.0,12.0,11.0,10.0,10.0,4*9.0,3*8.0,
     *                2*7.0,2*6.5,20*6.5,36*0.0/

      data snglntv /.75,.76,.76,7*.76,3*.755,
     *              6*.755,2*.755,3*.745,.735,5*.725,5*.715,.70,
     *              0.89,0.89,8*0.93,3*0.95,23*1.00/
      data snglntvcl /.76,.77,.78,7*.80,3*.80,
     *                6*.80,16*.80,.80,
     *                .82,.82,8*.85,3*.85,23*.90/
      data snglntvch /.72,.73,.72,7*.72,3*.71,
     *                6*.70,2*.68,3*.67,.66,10*.65,.60,
     *                0.96,0.96,8*1.00,3*1.05,23*1.10/

      data snglntd /-29.5,-28.0,-26.5,-26.5,-25.5,-25.0,
     *              -23.00,-22.00,-21.00,-20.5,-20.00,-19.5,
     *              -18.00,-16.00,-15.50,-14.50,-14.00,-14.00,
     *              -13.00,-12.50,-12.5,-12.5,-12.5,-10.0,
     *              2*-10.0,2*-10.0,8*-8.00,36*0.0/
      data snglntdcl /-34.0,-33.0,2*-32.0,-31.0,-30.0,
     *                -29.0,-28.0,-27.0,-26.0,-26.0,-25.0,
     *                -24.0,-22.0,-20.0,-18.0,-18.0,-17.0,
     *                -17.0,-17.0,-17.0,-16.0,-16.0,-12.0,
     *                2*-12.0,2*-12.0,8*-10.0,36*0.0/
      data snglntdch /-25.0,-23.0,-21.0,-21.0,-20.0,-20.0,
     *                -19.0,-18.0,-17.0,-17.0,-16.0,-16.0,
     *                -14.0,-14.0,-13.0,-13.0,-12.0,-11.0,
     *                -11.0,-10.0,-10.0,-9.0,-9.0,-8.0,
     *                2*-8.0,2*-8.0,8*-6.0,36*0.0/
```

```
c    Include file for initializing thresholds used in
c    the cloud mask algorithm.
c

      real dobt11(4),do11_12lo(4),dovrathi(4),
     *     dovratlo(4),doref2(4),doref3(4),
     *     do11_4lo(4),doco2(4),do11_12hi(1)

      real snglnt2(36,2),snglnt2cl(36,2),snglnt2ch(36,2),
     *     snglntv(36,2),snglntvcl(36,2),snglntvch(36,2)

      real nobt11(4),no11_12lo(4),no11_4lo(4),
     *     noco2(4)

c     real dlvrathi(4),dlvratlo(4),
      real dlvrat(4),
     *     dlref1(4),dlref3(4),dl11_4lo(4),
     *     dlco2(4),dl11_12hi(1),dovar11(1)

      real nl4_12hi(4),nl11_4lo(4),nlco2(4)

      real pds4_11(4),pdsco2(4),pdsref3(4)

      real pns4_12hi(4),pns11_4lo(4),pnsco2(4)

      real ds11_12hi(4),dsvrathi(4),dsvratlo(4),dsref2(4),
     *     ds11_4lo(4),ds11_4hi(4),dsco2(4),dsref3(4)

c     The four parameters represent:
c        1 - the low clear confidence threshold (0 confidence)
c        2 - the actual straight threshold value (50% confidence)
c            (pass/no pass)
c        3 - the high clear confidence threshold (100% confidence)
c        4 - the s curve power used to interpolate confidences
c
c    ** exceptions are the default thin cirrus 11-12 bt threshold
c       and the spatial variability thres. which currently need
c       only one value **

c    ** also the ocean sunglint thresholds which are dimensioned
c       36 (reflectance angles) X 2 for mid-point, high confidence,
c       and low confidence values.

c**** Day time ocean thresholds ********************
      data dobt11    /267.0, 270.0, 273.0, 1.0/
      data do11_12lo /0.05, 0.20, 0.35, 1.0/
      data do11_12hi /3.0/
      data dovrathi  /0.95, 1.10, 1.15, 1.0/
      data dovratlo  /0.95, 0.90, 0.85, 1.0/
      data doref2    /8.00, 7.00, 6.50, 1.0/
      data doref3    /4.00, 3.50, 3.00, 1.0/
      data do11_4lo  /-10.0, -8.0, -6.0, 1.0/
      data doco2     /219.0, 220.0, 221.0, 1.0/
      data dovar11   /0.50/

c**** Ocean sunglint thresholds.

      data snglnt2 /19.5,18.5,30.0,30.0,30.0,30.0,30.0,30.0,
     *              30.0,30.0,25.0,25.0,25.0,25.0,25.0,25.0,
     *              20*10.0,36*0.0/
      data snglnt2cl /25.0,25.0,40.0,60.0,60.0,60.0,50.0,50.0,
     *               50.0,50.0,6*35.0,20*13.5,36*0.0/
      data snglnt2ch /14.0,12.0,11.0,10.0,10.0,4*9.0,3*8.0,
     *               2*7.0,2*6.5,20*6.5,36*0.0/

      data snglntv /.95,.96,.96,7*.96,3*.955,
     *              6*.955,2*.955,3*.945,.935,5*.925,5*.915,.90,
     *              0.99,0.99,8*1.03,3*1.05,23*1.10/

      data snglntvcl /.96,.97,.98,7*.98,3*.97,
     *               6*.98,16*.96,.95,
     *               .92,.92,8*.90,3*.95,23*.95/
      data snglntvch /.92,.93,.92,7*.92,3*.91,
     *               6*.90,2*.88,3*.87,.86,10*.85,.85,
     *               1.03,1.03,8*1.10,3*1.10,23*1.15/

c**** Night time ocean thresholds ********************
      data nobt11    /267.0, 270.0, 273.0, 1.0/
      data no11_12lo /0.05, 0.20, 0.35, 1.0/
      data no11_4lo  /0.70, 0.60, 0.50, 1.0/
      data noco2     /219.0, 220.0, 221.0, 1.0/

c**** Day time Land thresholds ************************
c     data dlvrathi  /0.95, 1.10, 1.15, 1.0/
c     data dlvratlo  /0.95, 0.90, 0.85, 1.0/
      data dlvrat    /1.85, 1.90, 1.95, 1.0/
c     data dlref1    /18.0, 16.0, 14.0, 1.0/
      data dlref1    /29.0, 27.0, 25.0, 1.0/
      data dlref3    /4.00, 3.50, 3.00, 1.0/
c     data dlref3    /3.00, 2.50, 2.00, 1.0/
c     data dl11_4lo  /-14.0, -12.0, -10.0, 1.0/
      data dl11_4lo  /-18.0, -16.0, -14.0, 1.0/
      data dl11_12hi /3.0/
      data dlco2     /219.0, 220.0, 221.0, 1.0/

c**** Night time Land thresholds **********************
      data nl4_12hi  /4.50, 4.00, 3.50, 1.0/
      data nl11_4lo  /0.70, 0.60, 0.50, 1.0/
      data nlco2     /219.0, 220.0, 221.0, 1.0/

c**** Polar day time thresholds ***********************
      data pdsco2    /219.0, 220.0, 221.0, 1.0/
      data pds4_11   /11.00, 9.00, 7.00, 1.0/
      data pdsref3   /4.00, 3.50, 3.00, 1.0/

c**** Polar Night time thresholds *********************
      data pns4_12hi /4.50, 4.00, 3.50, 1.0/
      data pns11_4lo /0.70, 0.60, 0.50, 1.0/
      data pnsco2    /219.0, 220.0, 221.0, 1.0/

c**** Desert Thresholds *******************************
      data ds11_12hi /3.5, 3.0, 2.5, 1.00/
c     data dsvrathi  /1.10, 1.05, 1.00, 1.00/
c     data dsvratlo  /0.80, 0.85, 0.90, 1.00/
      data dsref2    /34.0, 30.0, 26.0, 1.00/
      data ds11_4lo  /-20.0, -18.0, -16.0, 1.00/
      data ds11_4hi  /-1.00, -3.00, -5.00, 1.00/
      data dsco2     /219.0, 220.0, 221.0, 1.0/
      data dsref3    /4.0, 3.5, 3.0, 1.0/
```

```
        nptests = nptests + 1
      end if
      loconf = dfthrsh + 0.5
      hiconf = dfthrsh - 0.5
      call conf_test(masdf1,loconf,hiconf,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir11 .le. 320.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.ds11_4lo(2) .or. mas11_4.le. ds11_4hi(2)) then
            nptests = nptests + 1
          end if
          locut(1) = ds11_4lo(1)
          locut(2) = ds11_4hi(1)
          hicut(1) = ds11_4lo(3)
          hicut(2) = ds11_4hi(3)
          midpt(1) = ds11_4lo(2)
          midpt(2) = ds11_4hi(2)

          call conf_test(mas11_4,locut,hicut,1.0,midpt,2,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        endif
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******      END OF GROUP 2 TESTS **************************
c
c
c
c ********  START OF GROUP 3 TESTS **************************

      nptests = 0

c ... visible (.88 micron) reflectance threshold test.
      if (visusd) then
        if (masv88.le.dsref2(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv88,dsref2(1),dsref2(3),dsref2(4),
     +                 dsref2(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS   **************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS  ************************

      nptests = 0

c ... near infrared high cloud test
      if (visusd) then
        if (masv188.le.dsref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dsref3(1),dsref3(3),dsref3(4),
     +                 dsref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ***********   END OF GROUP 4 TESTS   **************************

c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine LandDay_c(nbands,pxldat,avgtherm,vza,visusd,
     +                     vrused,eco_type,testbits,confdnc)

c F77 ********************************************************
c
c      Routine for performing clear sky tests over coastal regions
c      during daylight hours.
c
c Input parameters:
c nbands         Number of MAS channels.
c pxldat         Array containing reflectance or brightness temperatures
c avgtherm       Average brightness temperature for given box
c                for all bands for a single pixel
c vza            Current pixel viewing angle
c visusd         Logical variable indicating whether vis data used or not
c eco_type       Holder of ecosystem type (1-17)
c Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c END**********************************************************
c
      include 'thresholds.inc'

c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      logical visusd,vrused
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masir8,masv162,masv188,
     +     masir3,masv95,masv66,masv88,pi,schi,vrat,c6,pre_confdnc,
     +     groups,fac
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied in each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv95 = pxldat(4)
      masv162 = pxldat(5)
      masv188 = pxldat(6)
      masir3 = pxldat(8)
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent test group confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... perform tests.
c
C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.dlco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dlco2(1),dlco2(3),dlco2(4),
     *               dlco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  **************************
c
c
c
c     ****  GROUP 2 TESTS  *********************************

      nptests = 0

c ... 11-12um brightness temperature difference test (APOLLO test)
c ... for thin cirrus).
      masdf1 = masir11 - masir12
c ... calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... if a valid threshold was determined, then use this
c ... value, otherwise use the standard threshold
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = dl11_12hi(1)
      else
        dfthrsh = diftemp
      end if

      if (masdf1.le.dfthrsh) then
```

```
        nptests = nptests + 1
      end if
      locut(1) = dfthrsh + 0.5
      hicut(1) = dfthrsh - 1.25
      call conf_test(masdf1,locut,hicut,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir4.ne.32767.0.and.masir11.ne.32767.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.dl11_4lo(2)) then
            nptests = nptests + 1
          end if
          call conf_test(mas11_4,dl11_4lo(1),dl11_4lo(3),dl11_4lo(4),
     +                   dl11_4lo(2),1,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        end if
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******      END OF GROUP 2 TESTS **************************
c
c
c
c ********  START OF GROUP 3 TESTS **************************

      nptests = 0

c ... visible reflectance threshold test.
      if (visusd) then
        if (masv66.le.dlref1(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv66,dlref1(1),dlref1(3),dlref1(4),
     +                 dlref1(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS    **************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS  ***********************

      nptests = 0

c ... near infrared high cloud test
      if(visusd) then
        if (masv188.le.dlref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dlref3(1),dlref3(3),dlref3(4),
     +                 dlref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ***********   END OF GROUP 4 TESTS   **************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine LandDay_type1(nbands,pxldat,avgtherm,vza,visusd,
     +                         vrused,eco_type,testbits,confdnc)

c F77 ********************************************************
c
c      Routine for performing clear sky tests over land
c      surfaces during daylight hours.
c
c Input parameters:
c nbands         Number of MAS channels.
c pxldat         Array containing reflectance or brightness temperatures
c avgtherm       Average brightness temperature for given box
c                for all bands for a single pixel
c vza            Current pixel viewing angle
c visusd         Logical variable indicating whether vis data used
c vrused         Logical variable indicating if vis ratio test used
c eco_type       Holder of ecosystem type (1-17)
c Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c END************************************************************
c
      include 'thresholds.inc'

c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      logical visusd,vrused
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masir8,masv162,masv188,
     +     masir3,masv95,masv66,masv88,pi,schi,vrat,c6,pre_confdnc,
     +     groups,fac,masdf2,c7,tri_thres
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
      integer ngtests(4)
c ...
c ... external subroutines ..
      external conf_test,tview,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied in each test group.
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv95 = pxldat(4)
      masv162 = pxldat(5)
      masv188 = pxldat(6)
      masir3 = pxldat(8)
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)
      masdf1 = masir11 - masir12
      masdf2 = masir8 - masir11

c ... The "cmin" variables represent test group confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

c ... Perform tests.
c
C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... Co2 high cloud test.
      if (masir13.gt.dlco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,dlco2(1),dlco2(3),dlco2(4),
     *               dlco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  *************************
c
c     ****  GROUP 2 TESTS  ********************************

      nptests = 0

c ... 11-12um brightness temperature difference test (APOLLO test)
c ... for thin cirrus.
      masdf1 = masir11 - masir12
c ... Calculate secant of viewing zenith angle.
      cosvza = cos(vza*dtr)
      if (cosvza.ne.0.0) then
        schi = 1.0/cosvza
      else
        schi = 99.0
      end if

c ... Interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
      call tview(1,schi,masir11,diftemp)

c ... If a valid threshold was determined, then use this
c ... value, otherwise use the standard threshold.
      if (diftemp.lt.0.1 .or. schi.eq.99.0) then
        dfthrsh = dl11_12hi(1)
      else
        dfthrsh = diftemp
      end if
```

```
      if (masdf1.le.dfthrsh) then
        nptests = nptests + 1
      end if
      locut(1) = dfthrsh + 0.5
      hicut(1) = dfthrsh - 1.25
      call conf_test(masdf1,locut,hicut,1.0,dfthrsh,1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        if(masir4.ne.32767.0.and.masir11.ne.32767.0) then
          mas11_4 = masir11 - masir4
          if (mas11_4.ge.dl11_4lo(2)) then
            nptests = nptests + 1
          end if
          call conf_test(mas11_4,dl11_4lo(1),dl11_4lo(3),dl11_4lo(4),
     +                   dl11_4lo(2),1,c3)
          cmin2 = min(cmin2,c3)
          ngtests(2) = ngtests(2) + 1
        end if
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS **************************
c
c
c
c ********  START OF GROUP 3 TESTS **************************

      nptests = 0

c ... Visible reflectance threshold test.
      if (visusd) then
        if (masv66.le.dlref1(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv66,dlref1(1),dlref1(3),dlref1(4),
     +                 dlref1(2),1,c4)
        cmin3 = min(cmin3,c4)
        ngtests(3) = ngtests(3) + 1
      end if

c ... Visible channel ratio test (use GEMI variant).
      if (visusd .and. vrused ) then
        etan = 2.0*(masv88-masv66) + 1.5*masv88 + 0.5*masv66
        etad = masv88 + masv66 + 0.5
        eta = etan / etad
        vrat=eta*(1.0-0.25*eta) - ((masv66-0.125)/(1.0-masv66))
        if(vrat .gt. dlvrat(2)) then
          nptests = nptests + 1
        end if
        call conf_test(vrat,dlvrat(1),dlvrat(3),dlvrat(4),
     +         dlvrat(2),1,c5)
        cmin3 = min(cmin3,c5)
        ngtests(3) = ngtests(3) + 1
      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS   **************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS  ************************

      nptests = 0

c ... Near-infrared high cloud test.
      if(visusd) then
        if (masv188.le.dlref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,dlref3(1),dlref3(3),dlref3(4),
     +                 dlref3(2),1,c6)
        cmin4 = min(cmin4,c6)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************   END OF GROUP 4 TESTS   **************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv66 .lt. dlref1(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     +                          testbits,confdnc)
C!F77 ********************************************************
c      Routine for performing clear sky tests over land
c      surfaces during nighttime hours.
c
c      A new algorithm is being tested where groups of tests
c      will be used.  The groupings represent spectral tests which
c      attempt to detect the same type of clouds.  The minimum
c      confidence result from all grouped tests will be used to
c      determine a group confidence which will then be multiplied
c      with other group results to determine a final pixel clear
c      confidence value.
c
c      For nighttime land the groups are:
c          Group 1: High thick cloud
c                   13.9 micron bt test (masir13)
c
c          Group 2: Low cloud - thick
c                   8-11 micron and 11-12 micron bt tests
c                   11-4 micron bt tests
c
c          Group 5: High cloud - thin
c                   3.7-12 micron bt test
c
c!Input parameters:
c nbands         Total number of MAS channels.
c pxldat         Array containing reflectance or brightness temperatures
c                for all bands for a single pixel
c vza            Current pixel viewing angle
c eco_type       Ecosystem type (1-17)
c avgtherm       Average brightness temperature over given box
c!Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c!END*************************************************************
c
      INCLUDE 'thresholds.inc'

c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,dtr,mas4_12,masir11,masir12,masir13,masir4,
     +     pi,c3,mas11_4,groups,fac,pre_confdnc
      integer nptests
c ...
c ... local arrays ..
      integer ngtests(5)
c ...
c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos
c ...
c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0
      ngtests(5) = 0

c ... confidence to 1.0 to begin with
      confdnc = 1.0

c ... place band values into individual variables for easy
c ... identification
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0
      cmin5 = 1.0

C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.nlco2(2)) then
        nptests = nptests + 1
      end if
c     write(*,'(1x,''masir13 '',4f8.2)') masir13,nlco2(2),
c    *       nlco2(1),nlco2(3)
      call conf_test(masir13,nlco2(1),nlco2(3),nlco2(4),
     *               nlco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  ************************
c
c     ****  GROUP 2 TESTS  *********************************

      nptests = 0

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      mas11_4 = masir11 - masir4
      if (mas11_4.le.nl11_4lo(2)) then
        nptests = nptests + 1
      end if
c     write(*,'(1x,''mas11_4: '',6f10.2)') mas11_4,nl11_4lo(2),
c    *      nl11_4lo(1),nl11_4lo(3),masir11,masir4
      call conf_test(mas11_4,nl11_4lo(1),nl11_4lo(3),nl11_4lo(4),
     +               nl11_4lo(2),1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS *************************

c *******    START OF GROUP 5 TESTS  ***********************
```

```
      nptests = 0

c ... 4-12um brightness temperature difference test
c ... for thin cirrus).
      mas4_12 = masir4 - masir12
      if (mas4_12.le.nl4_12hi(2)) then
        nptests = nptests + 1
      end if
c     write(*,'(1x,''mas4_12: '',6f10.2)') mas4_12,nl4_12hi(2),
c    *       nl4_12hi(1),nl4_12hi(3),masir4,masir12
      call conf_test(mas4_12,nl4_12hi(1),nl4_12hi(3),nl4_12hi(4),
     +              nl4_12hi(2),1,c3)
      cmin5 = min(cmin5,c3)
      ngtests(5) = ngtests(5) + 1

      if(nptests .eq. ngtests(5) .and. ngtests(5) .ne. 0) then
        call set_bit(testbits,14)
      end if

c ********    END OF GROUP 5 TESTS  ***************************

c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin5
      groups = 0.0
      do kk = 1,5
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     write(*,'(1x,''confdnc '',8f8.5)') c1,c2,c3,
c    +       cmin1,cmin2,cmin5,fac,confdnc

      return
      end
```

```
      subroutine PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,
     +                         avgtherm,testbits,confdnc)

c F77 ********************************************************
c
c      Routine for performing clear sky tests over snow or ice
c      surfaces during daylight hours.
c
c Input parameters:
c nbands         Total number of MAS channels.
c pxldat         Array containing reflectance or brightness temperatures
c                for all bands for a single pixel
c vza            Current pixel viewing angle
c visusd         Logical variable indicating whether vis data used or not
c eco_type       Holder of ecosystem type (1-17)
c avgtherm       Average brightness temperature for given box
c!Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c END************************************************************
c
      include 'thresholds.inc'
c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      logical visusd
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,cosvza,dfthrsh,diftemp,dtr,mas11_4,masdf1,
     +     masir11,masir12,masir13,masir4,masir8,masv162,masv188,
     +     masir3,masv95,masv66,masv88,pi,
     +     masv55,masir65,pre_confdnc,fac,groups
      integer nptests
c ...
c ... local arrays ..
      integer ngtests(4)

c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group.
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv188 = pxldat(6)
      masv88 = pxldat(3)
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent test group confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin4 = 1.0

c ... Perform tests.

c     *****  GROUP 1 TESTS  ********************************

      nptests = 0

c ... co2 high cloud test
      if (masir13.gt.pdsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,pdsco2(1),pdsco2(3),pdsco2(4),
     *               pdsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  ************************
c
c
c
c     ****  GROUP 2 TESTS  *********************************

      nptests = 0

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      if (visusd) then
        mas11_4 = masir4 - masir11
        if (mas11_4.le.pds4_11(2)) then
          nptests = nptests + 1
        end if
        call conf_test(mas11_4,pds4_11(1),pds4_11(3),pds4_11(4),
     +                 pds4_11(2),1,c2)
        cmin2 = min(cmin2,c2)
        ngtests(2) = ngtests(2) + 1
      end if

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS *************************
c
c
c
c
c ***********   START OF GROUP 4 TESTS  ********************

      nptests = 0

c ... Near-infrared high cloud test.
      if (visusd) then
        if (masv188.le.pdsref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,pdsref3(1),pdsref3(3),pdsref3(4),
     +                 pdsref3(2),1,c3)
        cmin4 = min(cmin4,c3)
        ngtests(4) = ngtests(4) + 1
```

```
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ************   END OF GROUP 4 TESTS   *************************
c
c
c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv88 .lt. doref2(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine PolarNite_snow(nbands,pxldat,vza,eco_type,
     +                          avgtherm,testbits,confdnc)

C F77 *********************************************************
c
c      Routine for performing clear sky tests over snow
c      surfaces during nighttime hours.
c
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Viewing zenith angle
c eco_type      Holder of ecosystem type (1-17)
c avgtherm      Average brightness temperature for given box
c!Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END*********************************************************
c
      include 'thresholds.inc'    '
c ...
c ... scalar arguments ..
      real confdnc,vza
      integer nbands
      byte eco_type
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,dtr,mas4_12,masir11,masir12,masir13,masir4,
     +     pi,c3,mas11_4,pre_confdnc,fac,groups
      integer nptests
c ...
c ... local arrays ..
      integer ngtests(5)

c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group.
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0
      ngtests(5) = 0

c ... Set confidence to 1.0.
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent group test confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin5 = 1.0

C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c ... Co2 high cloud test.
      if (masir13.gt.pnsco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,pnsco2(1),pnsco2(3),pnsco2(4),
     *               pnsco2(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  *************************
c
c
c     ****  GROUP 2 TESTS  ***********************************

      nptests = 0

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      mas11_4 = masir11 - masir4
      if (mas11_4.le.pns11_4lo(2)) then
        nptests = nptests + 1
      end if
      call conf_test(mas11_4,pns11_4lo(1),pns11_4lo(3),pns11_4lo(4),
     +               pns11_4lo(2),1,c2)
      cmin2 = min(cmin2,c2)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS ***************************
c
c
c *******    START OF GROUP 5 TESTS  ************************

      nptests = 0

c ... 4-12um brightness temperature difference test
c ... for thin cirrus.
      mas4_12 = masir4 - masir12
      if (mas4_12.le.pns4_12hi(2)) then
        nptests = nptests + 1
      end if
      call conf_test(mas4_12,pns4_12hi(1),pns4_12hi(3),pns4_12hi(4),
     +               pns4_12hi(2),1,c3)
      cmin5 = min(cmin5,c3)
      ngtests(5) = ngtests(5) + 1

      if(nptests .eq. ngtests(5) .and. ngtests(5) .ne. 0) then
        call set_bit(testbits,14)
      end if

c ********    END OF GROUP 5 TESTS  **************************
```

```
c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2 * cmin5
      groups = 0.0
      do kk = 1,5
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

      return
      end
```

```
      subroutine analysis_proc(mlin,ibes,nelet,inchns,nele,line,refang,
     *                         ibele,pcwatr,ecstypes,lst,jday,pw,
     *                         rlats,rlons,sza,vza,raz,
     *                         cwl,indat,scf,solirr,
     *                         npix,nlsm,n2sm,n3sm,n4sm,bitarray,outrec)

c*****************************************************************

      include 'cldmask.inc'

c*****************************************************************

      integer*2 indat(npixel,nbands)

      integer*4 mlin,ibes,nelet,nele,line,ibele,jday,outrec,
     *          lst(npixel,nlcntx),pcwatr(npixel,nlcntx),inchns(inband),
     *          rglst(nlcntx,necntx),npix,nlsm,n2sm,n3sm,n4sm,
     *          rgpcwt(nlcntx,necntx)

      real*4 pxldat(nbands),sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       raz(npixel,nlcntx),cwl(50),scf(50),solirr(50),
     *       refang(npixel,nlcntx),rdat(npixel,nlcntx,nbands),
     *       pw,avgtherm(nbands),rgdata(nlcntx,necntx,nbands),
     *       rglat(nlcntx,necntx),rglon(nlcntx,necntx),
     *       rgsza(nlcntx,necntx),rgvza(nlcntx,necntx),
     *       rgraz(nlcntx,necntx),rgrfa(nlcntx,necntx),
     *       rlats(npixel,nlcntx),rlons(npixel,nlcntx),
     *       diff(nlcntx*necntx-1,nbands)

      byte eco_type,testbits(2),bitarray(2,npixel),
     *     ecstypes(npixel,nlcntx),rgeco(nlcntx,necntx)

      logical*4 snglnt,polar,water,land,day,night,visusd,snow,
     *          shadow,uniform,ap_sea_ice,ap_snow,ice,coast,desert,
     *          vrused,cnlwc

      integer*4 outrec_wk
      data outrec_wk /1/
c*****************************************************************

c         Unscale brightness temperature values and convert visible radiances
c         to reflectances for necessary pixels on the current scan line.
          call get_data(npixel,nlcntx,ibes,nelet,mlin,inband,nbands,
     *                  inchns,cwl,indat,sza,vza,scf,solirr,rdat)

          if(mlin .eq. nlcntx) then

c           Data block has been filled. In this version, the context is
c           defined as ('nlcntx' * 'necntx') pixels centered on the current
c           pixel. This box "slides along" the data block as the pixel number
c           is incremented. Therefore, the number of contexts in a scan line
c           is the same as the number of pixels processed ('nele').

            nmcntx = nele

c           'jlin' is the current scan line being processed (middle scan
c           of the current data block).
c           'klin' is the first line in the current data block.

            jlin = line - ((nlcntx-1) / 2)
            klin = mlin - (nlcntx-1)
            if(mod(jlin,100) .eq. 0) then
              write(*,'(1x,''Processing data block at line '',i10)')
     *             jlin
            end if

c           'kele' is first pixel of the scan to be processed.
c           'jele' is first required pixel of the scan.
            kele = ibele
            jele = ibes

c*****************************************************************

c           Loop over the number of contexts in the current data block.

            do 200 nc = 1,nmcntx

c             Get ancillary data for all pixels in the current context (region)
c             and set logical flags.
              call reg_anc(npixel,nlcntx,necntx,jele,klin,rlats,
     *                     rlons,sza,vza,raz,refang,pcwatr,ecstypes,
     *                     rglat,rglon,rgsza,rgvza,rgraz,rgrfa,
     *                     rgpcwt,lst,rglst,rgeco,uniform,cnlwc)

c             Get reflectance and brightness temperature data for all pixels
c             in the current context.
              call reg_data(npixel,nlcntx,necntx,jele,klin,inband,
     *                      nbands,rdat,rgdata)

c             Get reflectance and BT values for current pixel.
              call get_pxldat(nlcntx,necntx,inband,nbands,rgdata,
     *                        pxldat)

c             Quick averaging of MAS BT's over context to reduce noise.
              call atherm(nlcntx,necntx,inband,nbands,inchns,rgdata,
     *                    avgtherm)

c             Get ancillary data and set logical flags for the current
c             pixel (middle pixel of current context).
              call pxl_anc(nlcntx,necntx,rgvza,rgsza,rgrfa,rgpcwt,rgeco,
     *                     rglst,rglat,pxvza,pxrfa,pxpcwt,eco_type,
     *                     day,night,land,water,polar,snglnt,cnlwc,
     *                     visusd,vrused,ap_sea_ice,ap_snow,snow,ice,
     *                     coast,desert,nbands,pxldat,avgtherm,pxsza,
     *                     jday)

c
c             Initialize (clear) output "bit" array.
              do iwd = 1,2
                testbits(iwd) = 0
              enddo

c             Set "Additional Information" and "spare" bits (value=1).
              call set_bit(testbits,7)
              call set_bit(testbits,8)
              call set_bit(testbits,9)
              call set_bit(testbits,15)

c*****************************************************************

c             Begin tests for cloud mask determination.

              if (polar .and. night) then

c               Polar nighttime processing.

                call polar_nite(pxldat,pxvza,eco_type,land,ice,snow,
     *                          uniform,pw,nlcntx,necntx,inband,nbands,
     *                          rgdata,avgtherm,diff,testbits,
     *                          confdnc)

              else if(coast) then

c               Coastal region.
```

```
               if(day) then
c                Daytime processing.
                 call coast_day(pxldat,pxvza,visusd,eco_type,
     .                          desert,snow,nlcntx,necntx,nbands,
     .                          avgtherm,testbits,confdnc)

               else
c                Nighttime processing.
                 call coast_nite(pxldat,pxvza,eco_type,snow,nlcntx,
     .                           necntx,nbands,avgtherm,testbits,
     .                           confdnc)

               end if

             else if (water) then

c              Primarily water surface.

               if (day) then
c                Daytime processing.
                 call water_day(pxldat,pxvza,pxrfa,snglnt,visusd,
     .                          ice,uniform,eco_type,pw,nlcntx,
     .                          necntx,inband,nbands,rgdata,
     .                          avgtherm,diff,testbits,confdnc)
               else
c                Nighttime processing.
                 call water_nite(pxldat,pxvza,eco_type,ice,uniform,
     .                           pw,nlcntx,necntx,inband,nbands,
     .                           rgdata,avgtherm,diff,
     .                           testbits,confdnc)
               end if

             else if(land) then

c              Process as land surface.

               if (day) then
c                Daytime processing.
                 call land_day(pxldat,pxvza,visusd,vrused,
     .                         eco_type,desert,snow,nlcntx,necntx,
     .                         nbands,avgtherm,testbits,confdnc)
               else
c                Nighttime processing.
                 call land_nite(pxldat,pxvza,eco_type,snow,nlcntx,
     .                          necntx,nbands,avgtherm,testbits,
     .                          confdnc)
               end if

             end if

c*************************************************************************

c            Test for shadows, if necessary.
             if(land.and.day.and..not.snow.and.confdnc.ge.0.66) then
               call shadows(nbands,pxldat,shadow,testbits)
             end if

c            Test for possible non-cloud obstruction.
             if(land .and. day .and. .not. polar .and. .not. snow) then
               call noncld_obs_chk(nbands,pxldat,pxsza,testbits)
             end if

c            Set bits which indicate processing path through algorithm.
             call proc_path(water,land,coast,desert,day,snglnt,
     .                      snow,ice,testbits)

c            Get cloud mask statistics.
             call get_stats(confdnc,npix,n1sm,n2sm,n3sm,n4sm)

c            Set cloud mask quality bit flags.
             call set_confdnc(confdnc,testbits)

             do 250 ibyte = 1,2
               bitarray(ibyte,kele) = testbits(ibyte)
 250         continue

c            Increment counters.
             kele = kele + 1
             jele = jele + 1

  200      continue

c*************************************************************************

c            Write bit flags to output file for current scan line.

             outrec_wk = outrec_wk + 1
             outrec    = outrec_wk

c*************************************************************************

c            If not processing in single-line mode, re-buffer data in order
c            to add the next scan line and make a new data block.

             if(nlcntx .gt. 1) then
               do 400 il = 2,nlcntx
                 do 500 ie = ibes,nelet
                   do 600 k = 1,inband
                     rdat(ie,il-1,k) = rdat(ie,il,k)
 600               continue
 500             continue
 400           continue
             end if

             mlin = nlcntx - 1

           endif

         return
         end
```

```
      subroutine atherm(nlcntx,necntx,inband,nbands,inchns,rgdata,
     *                  avgtherm)

C     Computes the average brightness temperature over the
c     given box size (context) to reduce the noise.

      integer*4 inchns(inband)
      real*4 avgtherm(nbands),rgdata(nlcntx,necntx,nbands)

      do 100 k = 1,inband

        n = inchns(k)

        if(n .gt. 25) then

          knt = 0
          sum = 0.0

          do 200 i = 1,nlcntx

            do 300 j = 1,necntx

              if(rgdata(i,j,k) .ne. 32767.0) then
                knt = knt + 1
                sum = sum + rgdata(i,j,k)
              end if

300         continue

200       continue

          if(knt .gt. 0) then
            avgtherm(k) = sum / float(knt)
          else
            avgtherm(k) = 32767.0
          end if

        end if

100   continue

      return
      end
```

```
      real function bright50(band,cwl,r)

c     r         radiance in Watts per square meter per steradian per wavenumber
c     band      MAS band number (26-50)
c     bright50  temperature in Kelvin

      parameter(h = 6.6260755e-34)
      parameter(c = 2.99792458e+8)
      parameter(rk = 1.380658e-23)
      parameter(c1 = 2.0d0 * h * c * c)
      parameter(c2 = h * c / rk)

      integer*4 band
      real*4 ti(25),ts(25)

      data ti /
     *   7.876993e-01,  8.203697e-01,  6.643460e-01,  5.973398e-01,
     *   4.859185e-01,  4.063021e-01,  3.754345e-01,  3.043327e-01,
     *   2.783363e-01,  2.122330e-01,  2.006351e-01,  1.820766e-01,
     *   1.528009e-01,  1.314414e-01,  1.208291e-01,  9.452466e-02,
     *   2.024190e-02, -4.009836e-02, -5.146342e-02, -6.196085e-02,
     *  -6.241275e-02, -5.677923e-02, -6.189410e-02, -7.487083e-02,
     *  -5.352456e-02/

      data ts /
     *   9.992109e-01,  9.991444e-01,  9.992756e-01,  9.993234e-01,
     *   9.994270e-01,  9.995031e-01,  9.995242e-01,  9.996009e-01,
     *   9.996225e-01,  9.997025e-01,  9.997097e-01,  9.997280e-01,
     *   9.997643e-01,  9.997908e-01,  9.998015e-01,  9.998395e-01,
     *   9.997897e-01,  9.998696e-01,  9.999445e-01,  9.999673e-01,
     *   1.000004e+00,  1.000021e+00,  1.000030e+00,  1.000044e+00,
     *   1.000037e+00/

      if(band .lt. 26 .or. band .gt. 50) then
        write(*,'(1x,''MAS IR band must be in the range 26 - 50'')')
        go to 100
      end if

      ws = 1.0e-06 * cwl
      rs = 1.0e06 * r
      tc = c2 / ( ws * alog( c1 / ( rs * ws**5 ) + 1.0e+0 ) )
      bright50 = ( tc - ti( band - 25 ) ) / ts( band - 25 )
c     if(band .eq. 49) then
c       write(*,'(1x,''bright50 '',i10,4f8.3)') band,cwl,r,tc,bright50
c     end if

  100 return
      end
```

```
      subroutine chk_cnstnc(nlcntx,necntx,rgvza,rgsza,rgrfa,rgpcwt,
     *                      rgeco,rglat,rglon,cnpcwt,cnlwc)

c     Routine which checks consistency of various ancillary data and
c     sets flags accordingly.

      integer*4 rgpcwt(nlcntx,necntx),pxpcwt
      real*4 rgvza(nlcntx,necntx),
     *       rgrfa(nlcntx,necntx),rgsza(nlcntx,necntx),
     *       rglat(nlcntx,necntx),rglon(nlcntx,necntx)
      logical*4 cnpcwt,cnlwc
      byte rgeco(nlcntx,necntx),eco_type

c     Initializations.
      nwater = 0
      ncoast = 0
      nland = 0
      itotal = nlcntx * necntx

c     Loop over each pixel position in the current context.

      do i = 1,nlcntx
        do j = 1,necntx

c         Check land/sea tag (% coverage of water in pixel).
          pxpcwt = rgpcwt(i,j)
          if(pxpcwt .eq. 100) then
            nwater = nwater + 1
          else if(pxpcwt .eq. 50) then
            ncoast = ncoast + 1
          else if(pxpcwt .eq. 0) then
            nland = nland + 1
          end if

        enddo
      enddo

      if( (nwater+ncoast) .eq. itotal) then
        cnlwc = .true.
      else if( (nland+ncoast) .eq. itotal) then
        cnlwc = .true.
      else
        cnlwc = .false.
      end if

      if(nwater .eq. itotal) then
        cnpcwt = .true.
      else
        cnpcwt = .false.
      end if

c     write(*,'(1x,''consistency '',2l10)') cnlwc,cnpcwt

      return
      end
```

```
      subroutine chk_input(ibls,ibes,nelin,nelet,maxlin,maxele,
     *                     nlcntx,necntx,irt_code)

c     Check consistency of input information with number of scanlines
c     and pixels available.

c     Initialize return code
      irt_code = 0

      if(ibls .lt. 1 .or. nelin .gt. maxlin) then
         write(*,'(1x,''Invalid starting line or too many lines'')')
         irt_code = -1
      end if

      if(ibes .lt. 1 .or. nelet .gt. maxele) then
         write(*,'(1x,''Invalid starting pixel or too many pixels'')')
         irt_code = -2
      end if

      return
      end
```

```
      subroutine clear_bit(testbits,bit_num)

c     Routine for clearing a single bit within the 48-bit
c     cloud mask output array.

      integer*4 bit_num
      byte testbits(2)

c     Determine which word (1-2) of the 1-byte array contains the
c     bit of interest.

      iword = (bit_num / 8) + 1

c     Determine the position of the bit within the current
c     8-bit segment (1-byte word).

      ipos = bit_num - ((iword-1) * 8)

      itest = testbits(iword)
      itest = ibclr(itest,ipos)

      testbits(iword) = itest

c     write(*,'(1x,''clear '',4i10)') iword,ipos,itest,testbits(iword)

      return
      end
```

```
      subroutine coast_day(pxldat,vza,visusd,eco_type,
     *                     desert,snow,nlcntx,necntx,
     *                     nbands,avgtherm,testbits,confdnc)

c     Routine for setting appropriate flags and processing path
c     for daytime observations over coastal areas.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 visusd,snow,desert
      byte eco_type,testbits(2)

      if(snow) then

c       Use tests for snow covered conditions.
        call PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,
     *                     avgtherm,testbits,confdnc)

      else if (desert) then

c       Desert or semi-desert ecosystems.
        call DesertDay_c(nbands,pxldat,vza,visusd,avgtherm,
     *                   testbits,confdnc)

      else

c       Use default land tests.
        call LandDay_c(nbands,pxldat,avgtherm,vza,visusd,
     *                 vrused,eco_type,testbits,confdnc)

      end if

      return
      end
```

```
      subroutine coast_nite(pxldat,vza,eco_type,snow,nlcntx,
     *                      necntx,nbands,avgtherm,testbits,
     *                      confdnc)

c     Routine for setting appropriate flags and processing path
c     for nighttime observations over coastal regions.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 snow
      byte eco_type,testbits(2)

      if(snow) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,avgtherm,
     *                      testbits,confdnc)

      else

        call LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     *                      testbits,confdnc)

      end if

      return
      end
```

```
      subroutine conf_test(val,locut,hicut,power,midpt,nmval,
     *                     conflev)

c     Routine for determining the level of confidence of a
c     particular clear sky test.  Input single threshold value
c     ('midpt') with associated confidence limits ('locut', 'hicut')
c     or two threshold values and associated limits which define
c     a range of values for a test.  "Low" and "high" cutoffs
c     refer to low or high confidence ends of an interval and
c     not necessarily to absolute value.  Routine calculates the
c     confidence based on an "S" function.  One may change the shape
c     of the function by changing 'power' and/or 'midpt'.

      integer*4 nmval
      real*4 alpha,gamma,power,val,locut(2),hicut(2),range,
     *       coeff,s1,midpt(2),c
      logical*4 flipped

      coeff = 2.0 ** (power - 1.0)

c     Check if testing a single threshold or a range of values.

      if(nmval .eq. 1) then

c        Single threshold.

         if(hicut(1) .gt. locut(1)) then
            gamma = hicut(1)
            alpha = locut(1)
            flipped = .false.
         else
            gamma = locut(1)
            alpha = hicut(1)
            flipped = .true.
         end if
         beta = midpt(1)

c        Check for value beyond function range.

         if(.not. flipped .and. val .gt. gamma) then
            c = 1.0

         else if(.not. flipped .and. val .lt. alpha) then
            c = 0.0

         else if(flipped .and. val .gt. gamma) then
            c = 0.0

         else if(flipped .and. val .lt. alpha) then
            c = 1.0

         else

c           Value is within the range of the function.

            if(val .le. beta) then

               range = 2.0 * (beta - alpha)
               s1 = (val - alpha) / range
               if(.not.flipped) c = coeff * s1**power
               if(flipped) c = 1.0 - (coeff * s1**power)

            else

               range = 2.0 * (beta - gamma)
               s1 = (val - gamma) / range
               if(.not.flipped) c = 1.0 - (coeff * s1**power)
               if(flipped) c = coeff * s1**power

            end if

         end if

      else if(nmval .eq. 2) then

c        Range.

         gamma1 = hicut(1)
         gamma2 = hicut(2)
         alpha1 = locut(1)
         alpha2 = locut(2)
         beta1 = midpt(1)
         beta2 = midpt(2)

c        Find if interval between inner cutoffs passes test
c        or fails.

         if( (alpha1-gamma1) .gt. 0.0) then

c           Inner region fails test.
c           Check for value beyond function range.

            if(val .gt. alpha1 .and. val .lt. alpha2) then

               c = 0.0

            else if(val .lt. gamma1 .or. val .gt. gamma2) then

               c = 1.0

            else if(val .le. alpha1) then

c              Value is within range of lower set of limits.

               if(val .ge. beta1) then
                  range = 2.0 * (beta1 - alpha1)
                  s1 = (val - alpha1) / range
                  c = coeff * s1**power
               else
                  range = 2.0 * (beta1 - gamma1)
                  s1 = abs(val - gamma1) / range
                  c = 1.0 - (coeff * s1**power)
               end if

            else

c              Value is within range of upper set of limits.

               if(val .le. beta2) then
                  range = 2.0 * (beta2 - alpha2)
                  s1 = (val - alpha2) / range
                  c = coeff * s1**power
               else
                  range = 2.0 * (beta2 - gamma2)
                  s1 = (val - gamma2) / range
                  c = 1.0 - (coeff * s1**power)
               end if

            end if

         else

c           Inner region passes test.
c           Check for value beyond function range.

            if(val .gt. gamma1 .and. val .lt. gamma2) then
```

```
      integer function dhr2hms(dechrs)

c     Function which converts time in decimal hours to hms format.

      real*4 dechrs

      ih = dechrs

      a = dechrs - float(ih)
      b = a * 60.0
      im = b

      c = b - float(im)
      is = nint(c * 60.0 )

      if(is .eq. 60) then
        im = im + 1
        is = 0
      end if

      if(im .eq. 60) then
        ih = ih + 1
        im = 0
      end if

      dhr2hms = ih*10000 + im*100 + is

      return
      end
```

```
      subroutine file_close(if_hdfid,tp_unit,eco_unit,
     *                      outf_unit)

c     Routine for closing all files.

      integer*4 tp_unit,eco_unit,outf_unit,if_hdfid,sfend

      close (tp_unit)
      close (outf_unit)
      close (eco_unit)
      irt = sfend(if_hdfid)

      return
      end
```

```
      subroutine file_open(in_file,out_file,eco_index,if_hdfid,
     *                     tp_unit,eco_unit,outf_unit,irt_code)

c     Routine for opening all files needed for processing.
c     Returns necessary unit numbers to calling routine.

      character*50 topog,eco_file1,eco_file2,in_file,out_file
      integer*4 tp_unit,eco_unit,outf_unit,tp_un,eco_un,
     *          outf_un,sfstart,eco_index

c-----------------------------------------------------------------------

      include 'cldmask.inc'

c-----------------------------------------------------------------------

c     Names of topography and ecosystem files.
      data topog /'1st1km.v3'/
      data eco_file1 /'naogel_01g.img'/
      data eco_file2 /'ecosystem.img'/

c-----------------------------------------------------------------------

c     Define unit numbers for ancillary and output files.
      data tp_un /30/ , eco_un /80/ , outf_un /70/

      include 'hdf.inc'

c     Initialize return code.
      irt_code = 0

c     Open files.

c     Topography
      open (tp_un,file=topog,status='old',iostat=irt,
     *     access='direct',form='unformatted',recl=11)
      if(irt .ne. 0) then
        write(*,'(1x,''Cannot open topography file on unit '',
     *            i5)') tp_un
        irt_code = irt
      end if

c     Ecosystems
      if(eco_index .eq. 1) then
c       North American 1 km
        open(eco_un,file=eco_file1,status='old',iostat=irt,
     *      access='direct',form='unformatted',recl=11329)
      else
c       Global 10 minute
        open(eco_un,file=eco_file2,status='old',iostat=irt,
     *      access='direct',form='unformatted',recl=2)
      end if
      if(irt .ne. 0) then
        write(*,'(1x,''Cannot open ecosystem file on unit '',
     *            i5)') eco_un
        irt_code = irt
      end if

c     Output binary cloud mask file (bit flags).
      write(*,'(1x,''Opening output file '',a50)') out_file
      open(outf_un,file=out_file,status='unknown',iostat=irt,
     *     access='direct',form='unformatted',recl=npixel*2)
      if(irt .ne. 0) then
        write(*,'(1x,''Cannot open output file on unit '',
     *            i5)') outf_un
        irt_code = irt
      end if

c     Open input HDF data file and initialize HDF software.

      if_hdfid = sfstart(in_file,DFACC_READ)

      tp_unit = tp_un
      eco_unit = eco_un
      outf_unit = outf_un

      if(irt_code .ne. 0) irt_code = -1

      return
      end
```

```
      subroutine get_data(npixel,nlcntx,spx,epx,mlin,inband,nbands,
     *                    inchns,cwl,indat,sza,vza,scf,
     *                    solirr,rdat)

c     Converts thermal channel radiances to brightness temperatures and
c     visible radiances to reflectances for the specified MAS
c     pixels on a scan line. Stores output values in arrays conforming
c     to a "data block" (see main program).

      integer*2 indat(npixel,nbands)
      integer*4 inchns(inband),spx,epx
      real*4 sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       rdat(npixel,nlcntx,nbands),scf(nbands),cwl(nbands),
     *       solirr(nbands)

      parameter(pi = 3.14159)
      parameter(dtr = pi/180.0)

c     Process scan line input data.

      do k = 1,inband

        n = inchns(k)

        do j = spx,epx

          szen = sza(j,mlin)
          vzen = vza(j,mlin)

          if(indat(j,k).ge.0.and.indat(j,k).ne.32767)then
            if(n .le. 25) then
              if (szen .lt. 85.0) then
                rad = indat(j,k) * scf(n)
                csza = cos(dtr*szen)
                refstor = (rad*pi) / (solirr(n)*csza)
                rdat(j,mlin,k) = refstor * 100.0
              else
                rdat(j,mlin,k) = 0.0
              endif
            else
              rad = indat(j,k) * scf(n)
              rdat(j,mlin,k) = bright50(n,cwl(n),rad)
            end if
          else
            rdat(j,mlin,k) = 32767.0
          end if

        enddo

      enddo


      return
      end
```

```
      subroutine get_geo(npixel,nlcntx,mdelel,mdele2,spx,
     *                   epx,mlin,rlat,rlon,rsza,rvza,rsolaz,
     *                   rsnsaz,rlats,rlons,sza,vza,raz,
     *                   refang)

c     Routine which returns relative azimuth angle and reflectance
c     angle for the specified MAS FOVs on a scan line.  Also stores
c     all necessary geometric data in arrays conforming to a "data
c     block" (geometry for all necessary pixels in each of 'nlcntx'
c     scan lines).  See main program.

      parameter(pi = 3.14159)
      parameter(dtr = pi/180.0)
      parameter(rtd = 180.0/pi)

      integer*4 spx,epx
      real*4 sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       raz(npixel,nlcntx),refang(npixel,nlcntx),
     *       rlats(npixel,nlcntx),rlons(npixel,nlcntx),
     *       rlat(npixel),rlon(npixel),rsza(npixel),rvza(npixel),
     *       rsolaz(npixel),rsnsaz(npixel)

c     Get sub-satellite latitude and longitude.

      slat1 = rlat(mdelel)
      slon1 = rlon(mdelel)
      slat2 = rlat(mdele2)
      slon2 = rlon(mdele2)
      sslat = (slat1 + slat2) / 2.0
      sslon = (slon1 + slon2) / 2.0

c     Loop through the needed FOVs.

      do npx = spx,epx

c       Get lat, lon, solar azimuth, sensor azimuth for current pixel.
        plat = rlat(npx)
        plon = rlon(npx)
        psolaz = rsolaz(npx)
        psnsaz = rsnsaz(npx)

c       Get relative azimuth angle (as defined by ERBE).
        rlaz = abs(180.0 - (abs(psnsaz - psolaz)))

c       Get solar reflectance angle for sun glint determination.

        vzar = rvza(npx)*dtr
        szar = rsza(npx)*dtr
        razr = rlaz*dtr
        cossna = sin(vzar)*sin(szar)*cos(razr) +
     *           cos(vzar)*cos(szar)
        rfa = acos(cossna) * rtd

        sza(npx,mlin) = rsza(npx)
        vza(npx,mlin) = rvza(npx)
        raz(npx,mlin) = rlaz
        refang(npx,mlin) = rfa
```

```
      subroutine get_pxldat(nlcntx,necntx,inband,nbands,
     *                      rgdata,pxldat)

c     Returns array of reflectances/brightness temperatures for
c     the current pixel.

      real*4 pxldat(nbands),rgdata(nlcntx,necntx,nbands)

      ie = necntx - ((necntx-1) / 2)
      il = nlcntx - ((nlcntx-1) / 2)

      do k = 1,inband
        pxldat(k) = rgdata(il,ie,k)
      enddo

      return
      end
```

```
      subroutine get_refl3(tbb3,tbb4,jday,sza,ref3)

c     Routine for calculating the reflectance of the
c     solar component of the MAS channel 3.75 um radiance.

      parameter(pi=3.14159)
      parameter(dtr=pi/180.0)
      parameter(c1=1.191e-16)
      parameter(c2=1.4388e-02)
      parameter(wvnm3=2671.65)
      parameter(wvnm4=907.28)
      parameter(wvln3p=10000.0/(wvnm3*1.0e+06))
      parameter(wvln3=10000.0/wvnm3)

      real*4 tbb3,tbb4,solr_irrad

c     Calculate thermal component of channel 3 radiance
c     based on channel 4 brightness temperature.

c     thr3 = (c1*(wvnm3**3)) / (exp(c2*wvnm3/tbb4)-1.0)
      thr3 = c1 / ( (wvln3p**5.0) * (exp(c2/(wvln3p*tbb4)) - 1.0) )
      thr3 = thr3 * 1.0e-06

c     Calculate total radiance of channel 3.

c     tot3 = (c1*(wvnm3**3)) / (exp(c2*wvnm3/tbb3)-1.0)
      tot3 = c1 / ( (wvln3p**5.0) * (exp(c2/(wvln3p*tbb3)) - 1.0) )
      tot3 = tot3 * 1.0e-06

c     Calculate solar irradiance for channel 3.

      solr = solr_irrad(wvln3,jday)
      solr = solr * cos(sza*dtr)

c     Get reflectance.

      sol3 = tot3 - thr3
      ref3 = ((sol3 * pi) / solr) * 100.0


      return
      end
```

```
      subroutine get_regdif(nlcntx,necntx,inband,nbands,
     *                      rgdata,diff)

c     Computes reflectance or brightness temperature differences between
c     the (center) pixel of interest and the surrounding pixel
c     values.

      real*4 diff(nbands,nlcntx*necntx-1),
     *       rgdata(nlcntx,necntx,nbands),a,b

      midi = nlcntx - ((nlcntx-1) / 2)
      midj = necntx - ((necntx-1) / 2)

      do k = 1,inband

        jj = 0
        do i = 1,nlcntx
          do j = 1,necntx

            if(midi .ne. i .or. midj .ne. j) then

              jj = jj + 1
              a = rgdata(i,j,k)
              b = rgdata(midi,midj,k)

              if(a .ne. 32767.0 .and. b .ne. 32767.0) then
                diff(k,jj) = a - b
              else
                diff(k,jj) = 32767.0
              end if

            end if

          enddo
        enddo
      enddo

      return
      end
```

```
      subroutine get_sfc(npixel,nlcntx,spx,epx,mlin,rlats,
     *                   rlons,tp_unit,eco_unit,eco_index,
     *                   ecstypes,lst,pcwatr)

c     Routine which returns land/sea tags and ecosystem types for
c     the specified pixels on a scan line. Also stores surface
c     data in arrays conforming to a "data block" (see main program).

      integer*4 tp_unit,eco_unit,pcwatr(npixel,nlcntx),spx,epx,
     *          eco_index,lst(npixel,nlcntx)
      real*4 rlats(npixel,nlcntx),rlons(npixel,nlcntx)
      real*8 dlat,dlon,line,samp,dlat1,dlon2
      byte map(11),ecorec_g(2),ecorec_1(11329),ecstypes(npixel,nlcntx)

      save nrec

      data nrec /0/

c     Loop through pixels of interest.

      do npx = spx,epx

c       Get latitude and longitude of current pixel.
        rlat = rlats(npx,mlin)
        rlon = rlons(npx,mlin)

c       Get land/sea flag (use 1 km map).
c
c       Get map coordinates corresponding to the lat, lon position.
c
        dlat = rlat
        dlon = rlon
        dlat1 = rlat
        dlon2 = rlon
        call getcoord(dlat,dlon,line,samp)
        irec = nint(line)
        ismp = nint(samp)
c       Read land/sea tag file.
        ibbyte = ((irec-1)*40031) + (ismp-1)

        newlin = ibbyte / 11 + 1
        newele = mod(ibbyte,11)
        if(mod(ibbyte,11) .eq. 0) then
          newlin = newlin - 1
          newele = 11
        end if
        read(tp_unit,rec=newlin) map
        lsf = map(newele)
        lst(npx,mlin) = lsf
        if(lsf .eq. 1 .or. lsf .eq. 4) then
          pcwatr(npx,mlin) = 0
        else if(lsf .eq. 2) then
          pcwatr(npx,mlin) = 50
        else
          pcwatr(npx,mlin) = 100
        end if

c       Get ecosystem type.

        if(eco_index .ne. 1) then

c         Use global 10 minute file.

c         Calculate ecosystem file record number using standard
c         longitude.
          lat_indx = nint((90.0 - rlat) * 6.0) + 1
          lon_indx = nint((rlon+180.0) * 6.0) + 1
          if(lat_indx .gt. 1080) lat_indx = 1080
          if(lon_indx .gt. 2160) lon_indx = 2160
          norec = ((lat_indx-1) * 2160) + lon_indx
          newlin = norec / 2 + 1
          newele = mod(norec,2)
          if(mod(norec,2) .eq. 0) then
            newlin = newlin - 1
            newele = 2
          end if
c         Read ecosystem file.
          read(eco_unit,rec=newlin) ecorec_g
          ecstypes(npx,mlin) = ecorec_g(newele)

        else

c         Get type from 1 km resolution N. American file.

          call nacoord(dlat1,dlon2,line,samp)
          irec = nint(line)
          ismp = nint(samp)
          if(irec .ne. nrec) then
            read(eco_unit,rec=irec) ecorec_1
          end if
          ecstypes(npx,mlin) = ecorec_1(ismp)
          nrec = irec

        end if

      enddo

      return
      end
```

```
      subroutine get_sfc(npixel,nlcntx,spx,epx,mlin,rlats,
     *                   rlons,tp_unit,eco_index,eco_unit,
     *                   ecstypes,lst,pcwatr)

c     Routine which returns land/sea tags and ecosystem types for
c     the specified pixels on a scan line. Also stores surface
c     data in arrays conforming to a "data block" (see main program).

      integer*4 tp_unit,eco_unit,pcwatr(npixel,nlcntx),spx,epx,
     *          eco_index,lst(npixel,nlcntx)
      real*4 rlats(npixel,nlcntx),rlons(npixel,nlcntx)
      real*8 dlat,dlon,line,samp,dlat1,dlon2
      byte map(11),ecorec_g(2),ecorec_1(11329),ecstypes(npixel,nlcntx)

      save nrec

      data nrec /0/

c     Loop through pixels of interest.

c      write(*,'(1x,''mlin,irec,nrec '',i8)') mlin
         write(6,*) 'spx=',spx,'    epx=',epx
         write(6,*) 'rlat=',rlat,'    rlon=',rlon
      do npx = spx,epx

c       Get latitude and longitude of current pixel.
        rlat = rlats(npx,mlin)
        rlon = rlons(npx,mlin)

c       Get land/sea flag (use 1 km map).
c
c       Get map coordinates corresponding to the lat, lon position.
c
        dlat = rlat
        dlon = rlon
        dlat1 = rlat
        dlon2 = rlon
        call getcoord(dlat,dlon,line,samp)
        irec = nint(line)
        ismp = nint(samp)
c       Read land/sea tag file.
        ibbyte = ((irec-1)*40031) + (ismp-1)

        newlin = ibbyte / 11 + 1
        newele = mod(ibbyte,11)
        if(mod(ibbyte,11) .eq. 0) then
          newlin = newlin - 1
          newele = 11
        end if
        read(tp_unit,rec=newlin) map
        lsf = map(newele)
        lst(npx,mlin) = lsf
        if(lsf .eq. 1 .or. lsf .eq. 4) then
          pcwatr(npx,mlin) = 0
        else if(lsf .eq. 2) then
          pcwatr(npx,mlin) = 50
        else
          pcwatr(npx,mlin) = 100
        end if

c       Get ecosystem type.

        if(eco_index .ne. 1) then

c         Use global 10 minute file.

c         Calculate ecosystem file record number using standard
c         longitude.

          lat_indx = nint((90.0 - rlat) * 6.0) + 1
          lon_indx = nint((rlon+180.0) * 6.0) + 1
          if(lat_indx .gt. 1080) lat_indx = 1080
          if(lon_indx .gt. 2160) lon_indx = 2160
          norec = ((lat_indx-1) * 2160) + lon_indx
          newlin = norec / 2 + 1
          newele = mod(norec,2)
          if(mod(norec,2) .eq. 0) then
            newlin = newlin - 1
            newele = 2
          end if
c         Read ecosystem file.
          read(eco_unit,rec=newlin) ecorec_g
          ecstypes(npx,mlin) = ecorec_g(newele)

        else

c         Get type from 1 km resolution N. American file.

          write(6,*) 'dlat1=',dlat1,'    dlon2=',dlon2
          call nacoord(dlat1,dlon2,line,samp)
          irec = nint(line)
          ismp = nint(samp)
          write(6,*) 'irec=',irec
          if(irec .ne. nrec) then
c             irec = 7900
            read(eco_unit,rec=irec) ecorec_1
c             write(*,'(1x,''clear!!! '')')
          end if
          ecstypes(npx,mlin) = ecorec_1(ismp)
          nrec = irec

        end if

      enddo

      return
      end
```

```
      subroutine get_stats(confdnc,npix,n1sm,n2sm,n3sm,n4sm)

c     Routine for counting frequencies of various confidence
c     categories.

      data nmpix /0/ , n1s /0/ , n2s /0/ , n3s /0/ , n4s /0/

      nmpix = nmpix + 1
      if(confdnc .gt. 0.99) then
        n1s = n1s + 1
      else if(confdnc .gt. 0.95) then
        n2s = n2s + 1
      else if(confdnc .gt. 0.66) then
        n3s = n3s + 1
      else if(confdnc .lt. 0.01) then
        n4s = n4s + 1
      end if

      npix = nmpix
      n1sm = n1s
      n2sm = n2s
      n3sm = n3s
      n4sm = n4s

      return
      end
```

```
      subroutine getsctm(nsec,bdate,iyrday,ihms,ihr,imin,isec)

c     Routine for converting seconds to hours, minutes, seconds.
c     Also checks for new day.

      integer*4 bdate

      iyr = bdate / 1000
      iday = mod(bdate,1000)

      ihr = nsec / 3600
      n1 = nsec - 3600*ihr
      imin = n1 / 60
      n2 = n1 - 60*imin
      isec = n2

      if(ihr .ge. 24) then
        ihr = ihr - 24
        iday = iday + 1
        leap = mod(iyr,4)
        if(iday .eq. 366) then
          if(leap .ne. 0) then
            iyr = iyr + 1
            iday = 1
          end if
        else if(iday .eq. 367) then
          iyr = iyr + 1
          iday = 1
        end if
      end if

      iyrday = iyr*1000 + iday
      ihms = ihr*10000 + imin*100 + isec

      return
      end
```

```
      integer function init_proc(ibele,nele,if_hdfid,tp_unit,eco_unit,
     *                     outf_unit,ibhms,iyrday,jday,maxlin,ibls,
     *                     ibes,nelin,nelet,procflg,pw,cwl,scf,solirr,
     *                     start,edge,gstart,gedge,eco_index,dataid,
     *                     latid,lonid,szaid,vzaid,solazid,snsazid,
     *                     sltid,block,
     *                     in_file,len1,out_file,len2,ibl,nlin)

c-----------------------------------------------------------------------

      include 'cldmask.inc'

c-----------------------------------------------------------------------
c
c       external variables

      integer*4 ibele            ! start pixel (run parameter)
      integer*4 nele             ! # of processing pixels (run parameter)
      integer*4 if_hdfid         ! ID No. of image data file
      integer*4 tp_unit          ! unit No. of land/sea tag file
      integer*4 eco_unit         ! unit No. of ecosystem file
      integer*4 outf_unit        ! unit No. of cloud flag file
      integer*4 ibhms            ! time of first scan line (hhmmss)
      integer*4 iyrday           ! date of first scan line (ddd)
      integer*4 jday             ! date of first scan line (ddd)
      integer*4 maxlin  ! number of lines in the image data file
      integer*4 ibls             ! start line No.
      integer*4 ibes             ! start pixel No.
      integer*4 nelin            ! end line No.
      integer*4 nelet            ! end pixel No.
      integer*4 procflg          ! procedure flag
      real*4 pw         ! precipitable water value (run parameter)
      real*4 cwl(50)             ! channel central wavelength
      real*4 scf(50)             ! scale factor for radiance data
      real*4 solirr(50)          ! spectral solar irradiance

      integer*4 start(3)     ! start pixel/line to read from scanline
      integer*4 edge(3)      ! edge pixel/line to read from scanline
      integer*4 gstart(3)    ! start pixel/line to read from geometry
      integer*4 gedge(3)     ! edge pixel/line to read from geometry
      integer*4 eco_index    ! ecosystem file index (run parameter)
      integer*4 dataid       ! radiance data ID
      integer*4 lonid        ! longitude ID
      integer*4 szaid        ! solar zenith angle ID
      integer*4 vzaid        ! viewing zenith angle ID
      integer*4 solazid      ! solar azimuth ID
      integer*4 snsazid      ! sensor azimuth ID
      integer*4 sltid        ! scan line time ID
      integer*4 block        ! block = (nelin-ibls)/linmax

      character*50 in_file   ! input file name (run parameter)
      integer*4 len1         ! length of input file name
      character*50 out_file  ! output file name (run parameter)
      integer*4 len2         ! length of output file name
      integer*4 ibl          ! start line No.(run parameter)
      integer*4 nlin         ! # of processing lines (run parameter)

c-----------------------------------------------------------------------
c
c       internal variables

c$$$      character*50 parm(8)
      character*72 sds_name

      integer*4 slcid,slymdid,cwlid,solirid,
     *          stride(3),gstride(3),atr_indx,dims(3),
     *          sffattr,sfrattr,sfrdata,sfginfo,dhr2hms
      integer*4 num_in,irt_code,irt,idbymd,dbtime,
     *          iyr,imo,ida,nda,numday,rank,num_type,nattrs

      data stride /1,1,1/
      data gstride /1,1,1/

c-----------------------------------------------------------------------

      include 'hdf.inc'

c-----------------------------------------------------------------------

c-----------------------------------------------------------------------
c     for AVS
c-----------------------------------------------------------------------
c     Get input from keyboard.
c$$$      do num_in = 1,8
c$$$        call getarg(num_in,parm(num_in))
c$$$      enddo
c$$$
c$$$      read(parm(1),'(a50)') in_file
c$$$      read(parm(2),'(a50)') out_file
c$$$      read(parm(3),'(i10)') ibl
c$$$      read(parm(4),'(i10)') nlin
c$$$      read(parm(5),'(i10)') ibele
c$$$      read(parm(6),'(i10)') nele
c$$$      read(parm(7),'(i10)') eco_index
c$$$      read(parm(8),'(f10.5)') pw

      in_file(len1+1:len1+1) = char(0)
      out_file(len2+1:len2+1) = char(0)

c-----------------------------------------------------------------------

      if ((pw.lt.1.) .or. (pw.gt.11.)) then
        write (*,'('' tri-spectral technique will not be used '')')
      end if

c-----------------------------------------------------------------------

c     Open files and obtain corresponding id/unit numbers.

      call file_open(in_file,out_file,eco_index,if_hdfid,tp_unit,
     *               eco_unit,outf_unit,irt_code)
      if(irt_code .lt. 0) go to 1000

c-----------------------------------------------------------------------

c     Make HDF calls to retrieve all necessary HDF id numbers, data scale
c     factors, channel central wavelengths and spectral solar irradiance.

c     Get IDs.
      call set_HDF(if_hdfid,dataid,latid,lonid,szaid,vzaid,slcid,
     *             sltid,slymdid,cwlid,solirid,solazid,snsazid)

c     Get scale factors for radiance data.
      atr_indx = sffattr(dataid,'scale_factor')
      irt = sfrattr(dataid,atr_indx,scf)

c     Get channel central wavelengths.
      start(1) = 0
      edge(1) = 50
      irt = sfrdata(cwlid,start,stride,edge,cwl)

c     Get spectral solar irradiance for each channel.
      irt = sfrdata(solirid,start,stride,edge,solirr)

c-----------------------------------------------------------------------
```

```
c     Get date and time of first scan line in the data.

      start(1) = 0
      edge(1) = 1

c     Year, month, day from input file (yymmdd).
      irt = sfrdata(slymdid,start,stride,edge,idbymd)

c     Time in decimal hours from input file.
      irt = sfrdata(sltid,start,stride,edge,dbtime)

c     Convert decimal hours to hours, minutes, seconds (hhmmss).
      ibhms = dhr2hms(dbtime)

c     Convert date from "yymmdd" to "yyddd".
      iyr = idbymd / 10000
      imo = mod((idbymd/100),(iyr*100))
      ida = mod(idbymd,(iyr*10000+imo*100))
      nda = numday(iyr,imo,ida)
      iyrday = (mod(iyr,100)*1000) + nda
      jday = mod(iyrday,1000)

      write(*,'(1x,''Beginning year,month,day: '',i4,i3,i4)')iyr,imo,ida
      write(*,'(1x,''Beginning time in decimal hours: '',f10.5)') dbtime

c**********************************************************************

c     Get number of scanlines in the file.
      irt = sfginfo(dataid,sds_name,rank,dims,num_type,nattrs)
      maxlin = dims(3)
c      write(*,'(1x,''# lines in file: '',i10)') maxlin

c**********************************************************************

c     Define constants based on user inputs.

c     'ibls' is first line # needed
c     'nelin' is last line # needed
c     'ibes' is first element # needed
c     'nelet' is last element # needed on each scan line

      ibls = ibl - ((nlcntx-1) / 2)
      ibes = ibele - ((necntx-1) / 2)
      nelin = ibls + (nlcntx-1) + (nlin-1)
      nelet = ibes + (necntx-1) + (nele-1)

c**********************************************************************

c     Check user input for consistency with data file size.

      call chk_input(ibls,ibes,nelin,nelet,maxlin,in_recl,
     *               nlcntx,necntx,irt_code)
      if(irt_code .lt. 0) go to 1000

c**********************************************************************

      start(1) = 0
      edge(1) = in_recl
      edge(2) = 1
      edge(3) = 1
      gstart(1) = 0
      gedge(1) = in_recl
      gedge(2) = 1

      block = (nelin - ibls) / linmax

c         Normal return

c      write(*,'(1x,''# initialize end'')')
      procflg = 0
      return

 1000 continue

c          Error return

      write(*,'(1x,''# initialize abnormal end'')')
      procflg = -1
      return

      end
```

```
      subroutine land_day(pxldat,vza,visusd,vrused,eco_type,
     .                    desert,snow,nlcntx,necntx,nbands,
     .                    avgtherm,testbits,confdnc)

c     Routine for setting appropriate flags and processing path
c     for daytime observations over land surfaces.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 visusd,snow,desert,vrused
      byte eco_type,testbits(2)

      if(snow) then

c       Use tests for snow covered conditions.
        call PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,
     .                     avgtherm,testbits,confdnc)

      else if (desert) then

c       Desert or semi-desert ecosystems.
        call DesertDay(nbands,pxldat,vza,visusd,avgtherm,
     .                 testbits,confdnc)

      else

c       Use default land tests.
        call LandDay_type1(nbands,pxldat,avgtherm,vza,visusd,
     .                     vrused,eco_type,testbits,confdnc)

      end if

      return
      end
```

```
      subroutine land_nite(pxldat,vza,eco_type,snow,nlcntx,
     *                     necntx,nbands,avgtherm,testbits,
     *                     confdnc)

c     Routine for setting appropriate flags and processing path
c     for nighttime observations over land surfaces.

      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands)
      logical*4 snow
      byte eco_type,testbits(2)

      if(snow) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,avgtherm,
     *                      testbits,confdnc)

      else

        call LandNite_type1(nbands;pxldat,avgtherm,vza,eco_type,
     *                      testbits,confdnc)

      end if

      return
      end
```

```
      integer function mainlp_proc(ibls,nelin,ibes,nelet,inchns,refang,
     *                     nele,maxlin,ibele,pcwatr,ecstypes,lst,jday,
     *                     pw,rlats,rlons,sza,vza,raz,indat,cwl,
     *                     scf,solirr,npix,n1sm,n2sm,n3sm,n4sm,
     *                     bitarray,outrec,block,count)
c**********************************************************************

      include 'cldmask.inc'

c**********************************************************************
c
c       external variables

      integer*4 ibls             ! start line No.
      integer*4 nelin            ! end line No.
      integer*4 ibes             ! start pixel No.
      integer*4 nelet            ! end pixel No.
      integer*4 inchns(inband,linmax)          ! input channel
      real*4 refang(npixel,nlcntx,linmax)    ! solar reflectance angle
      integer*4 nele             ! # of processing pixels (run parameter)
      integer*4 maxlin  ! number of lines in the image data file
      integer*4 ibele            ! start pixel (run parameter)
      integer*4 pcwatr(npixel,nlcntx,linmax)     ! sea tag
      byte ecstypes(npixel,nlcntx,linmax)        ! ecosystem types
      integer*4 lst(npixel,nlcntx,linmax)        ! land tag
      integer*4 jday             ! date of first scan line (ddd)
      real*4 pw                  ! precipitable water value
      real*4 rlats(npixel,nlcntx,linmax)     ! latitude
      real*4 rlons(npixel,nlcntx,linmax)     ! longitude
      real*4 sza(npixel,nlcntx,linmax)   ! solar zenith angle
      real*4 vza(npixel,nlcntx,linmax)   ! viewing zenith angle
      real*4 raz(npixel,nlcntx,linmax)   ! relative azimuth angle
      integer*2 indat(npixel,nbands,linmax)      ! image data
      real*4 cwl(50)             ! channel central wavelength
      real*4 scf(50)             ! scale factor for radiance data
      real*4 solirr(50)          ! spectral solar irradiance
      integer*4 npix             ! # of total pixels
      integer*4 n1sm             ! # of pixels larger than 99%
      integer*4 n2sm             ! # of pixels larger than 95%
      integer*4 n3sm             ! # of pixels larger than 66%
      integer*4 n4sm             ! # of pixels larger than 1%
      byte bitarray(2,npixel,linmax)           ! cloud flag data
      integer*4 outrec(outmax)   ! output record No.
      integer*4 block            ! block = (nelin-ibls)/linmax
      integer*4 count            ! count[0,block]

c**********************************************************************
c
c       internal variables

      integer*4 cnt,line,sline,eline
      integer*4 mlin /0/

c**********************************************************************

c     Begin input scan line loop.

c     'mlin' counts # lines processed in current region (context).
c     'line' is current line #.

C      mlin = 0

      if(count.eq.0) then
         do i=1,outmax
            outrec(i) = 1
         enddo
      end if

      do i=1,2
         do j=1,npixel
            do k=1,linmax
               bitarray(i,j,k) = 0
            enddo
         enddo
      enddo

      cnt = 0
      sline = ibls + linmax * count
      if(count.eq.block) then
         eline = nelin
      else
         eline = sline + linmax - 1
      endif

C      Do 100 line=ibls,nelin
      Do 100 line=sline,eline

        if(line .le. maxlin) then

          mlin = mlin + 1
          cnt = cnt + 1

          call analysis_proc(mlin,ibes,nelet,inchns(1,cnt),
     *                       nele,line,refang(1,1,cnt),
     *                       ibele,pcwatr(1,1,cnt),
     *                       ecstypes(1,1,cnt),lst(1,1,cnt),jday,pw,
     *                       rlats(1,1,cnt),rlons(1,1,cnt),
     *                       sza(1,1,cnt),vza(1,1,cnt),
     *                       raz(1,1,cnt),cwl,indat(1,1,cnt),
     *                       scf,solirr,npix,n1sm,n2sm,n3sm,n4sm,
     *                       bitarray(1,1,cnt),outrec(line))

        end if

 100  continue

      return
      end
```

```
      subroutine noncld_obs_chk(nbands,pxldat,pxsza,testbits)

      real*4 pxldat(nbands)
      byte testbits(2)
      logical fire,smoke

c     Routine which checks for the possible presence of smoke.

c     Initializations.
      fire = .false.
      smoke = .false.
      ref21 = pxldat(7)
      ref66 = pxldat(2)
      t375 = pxldat(8)
      t11 = pxldat(11)
      tdif = t375 - t11
      a = 7.0 + (ref21 / 2.0)

c     First, check for fires (hot spots).
      if(t375 .gt. 350.0 .and. tdif .gt. 10.0) then
          fire = .true.
      end if

c     Test for thick smoke or fire. If found, clear bit #8.
      if(ref21 .lt. 20.0) then
        if((ref66 .gt. a) .or. fire) then
          smoke = .true.
          call clear_bit(testbits,7)
        end if
      end if

      return
      end
```

```
      function numday(iyr,imo,ida)

c     Calculates numeric day of the year given month and day of year.

      integer*4 idays(12),ildays(12)

      data idays /0,31,59,90,120,151,181,212,243,273,304,334/
      data ildays /0,31,60,91,121,152,182,213,244,274,305,335/

      if(mod(iyr,4) .ne. 0) then
        numday = idays(imo) + ida
      else
        numday = ildays(imo) + ida
      end if

      return
      end
```

```
      subroutine ocean_day(nbands,pxldat,avgtherm,vza,pxrfa,
     +                     snglnt,visusd,eco_type,pw,
     +                     testbits,confdnc)
c F77 *********************************************************
c
c     Routine for performing clear sky tests over water
c     surfaces during daylight hours.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c vza           Current pixel viewing angle
c snglnt        Logical variable flagging sunglint pixels
c visusd        Logical variable indicating whether vis data used or not
c eco_type      Holder of ecosystem type (1-17)
c pw            Amount of precipitable water at pixel site
c avgtherm      Average BT over box of pixels
c pxrfa         reflectance angle used for determining sunglint
c Output Parameters:
c testbits      two word 1-byte array containing bit results
c confdnc       product of all applied individual confidences
c
c END*************************************************************
c
      INCLUDE 'thresholds.inc'
c ...
c ... scalar arguments ..
      real confdnc,pw,vza
      integer nbands
      logical snglnt,visusd
c ...
c ... array arguments ..
      real pxldat(nbands),avgtherm(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,c5,c6,c7,cosvza,ct1,ct2,dfthrsh,diftemp,diftsp1,
     +     diftsp2,dtr,mas11_4,masdf1,masdf2,masir11,masir12,masir13,
     +     masir4,masir8,masir3,masv188,masv95,masv66,masv88,
     +     pi,schi,vrat,masir65,c8,c9,c10,
     +     c11,mas_swir,groups,fac,pre_confdnc
      integer nptests,ngtests(4)
      byte eco_type
c ...
c ... local arrays ..
      real hicut(2),locut(2),midpt(2)
c ...
c ... external functions ..
      real rega,regb
      external rega,regb
c ...
c ... external subroutines ..
      external conf_test,tview,clear_bit,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos,cos
c ...
c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0
      ngtests(3) = 0
      ngtests(4) = 0

c ... set confidence to 1.0 to begin with
      confdnc = 1.0

c ... Place band values into individual variables for easy
c ... identification.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv95 = pxldat(4)
      masv188 = pxldat(6)
      masir3 = pxldat(8)
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... The "cmin" variables represent test group confidences.
      cmin1 = 1.0
      cmin2 = 1.0
      cmin3 = 1.0
      cmin4 = 1.0

C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c     11 micron brigthness temperature threshold test
c ... compare to daytime ocean threshold, set bit if passed
      if (masir11 .ge. dobt11(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir11,dobt11(1),dobt11(3),dobt11(4),
     +               dobt11(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

c ... co2 high cloud test
      if (masir13.gt.doco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,doco2(1),doco2(3),doco2(4),
     +               doco2(2),1,c2)
      cmin1 = min(cmin1,c2)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  *************************
c
c
c     ****  GROUP 2 TESTS  ********************************

      nptests = 0

c ... Tri-spectral tests - apply only if a pw value exists for this
c.... pixel.
      if (pw .gt. 1.0) then

c ...   get dynamic clear sky thresholds based on pw
        diftsp2 = rega(pw)
        diftsp1 = regb(pw)
c ...   calculate 8 minus 11 and 11 minus 12 micron BTDIFs
        masdf2 = masir8 - masir11
        masdf1 = masir11 - masir12
```

```
c ...    if both BTDIF's pass the tests, then set bit
         if ((masdf2.lt.diftsp2) .and. (masdf1.gt.diftsp1)) then
           nptests = nptests + 1
         end if
         locut(1) = diftsp2 + .5
         hicut(1) = diftsp2 - .5
         call conf_test(masdf2,locut,hicut,1.0,diftsp2,1,ct1)
         locut(1) = diftsp1 - .5
         hicut(1) = diftsp1 + .5
         call conf_test(masdf1,locut,hicut,1.0,diftsp1,1,ct2)
         c3 = ct1*ct2
         cmin2 = min(cmin2,c3)
         ngtests(2) = ngtests(2) + 1
         c4 = 0.0

       else

c ...    If no pw value exists for this pixel, use the low cloud
c ...    11-12 micron BTDIF test.

         masdf1 = masir11 - masir12
         if (masdf1.gt.doll_12lo(2)) then
           nptests = nptests + 1
         end if
         call conf_test(masdf1,doll_12lo(1),doll_12lo(3),doll_12lo(4),
     +                  doll_12lo(2),1,c4)
         cmin2 = min(cmin2,c4)
         ngtests(2) = ngtests(2) + 1
         c3 = 0.0

       end if

c ... 11-12um brightness temperature difference test
c ... for thin cirrus.
c ... calculate secant of viewing zenith angle.
       cosvza = cos(vza*dtr)
       if (cosvza.ne.0.0) then
         schi = 1.0/cosvza
       else
         schi = 99.0
       end if

c ... Interpolate look-up table values of 11 - 12 micron bt
c ... difference thresholds (function of viewing zenith
c ... and 11 micron brightness temperature).
       call tview(1,schi,masir11,diftemp)

c ... If a threshold was determined, then use this value
c ... as the test threshold, otherwise use the standard threshold.
       if (diftemp.lt.0.1 .or. schi.eq.99.0) then
         dfthrsh = doll_12hi(1)
       else
         dfthrsh = diftemp
       end if

       if (masdf1.le.dfthrsh) then
         nptests = nptests + 1
       end if
       locut(1) = dfthrsh + 0.5
       hicut(1) = dfthrsh - 1.25
       call conf_test(masdf1,locut,hicut,1.0,dfthrsh,1,c5)
       cmin2 = min(cmin2,c5)
       ngtests(2) = ngtests(2) + 1

c ... 11 minus 4 micron BTDIF fog and low cloud test.
       if (visusd) then
         mas11_4 = masir11 - masir4
         if (mas11_4.ge.doll_4lo(2)) then
           nptests = nptests + 1
         end if
         call conf_test(mas11_4,doll_4lo(1),doll_4lo(3),doll_4lo(4),
     +                  doll_4lo(2),1,c6)
         cmin2 = min(cmin2,c6)
         ngtests(2) = ngtests(2) + 1
       end if

       if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
         call set_bit(testbits,11)
       end if


c *******     END OF GROUP 2 TESTS *************************
c
c
c ********  START OF GROUP 3 TESTS *************************

       nptests = 0

c ... visible reflectance threshold test.

       if (visusd) then

         if(snglnt) then
           ibin = int(pxrfa) + 1
           if(ibin .gt. 36) ibin = 36
           locut(1) = snglnt2cl(ibin,1)
           hicut(1) = snglnt2ch(ibin,1)
           midpt(1) = snglnt2(ibin,1)
         else
           locut(1) = doref2(1)
           hicut(1) = doref2(3)
           midpt(1) = doref2(2)
         end if

         if (masv88.le.midpt(1)) then
           nptests = nptests + 1
         end if
         call conf_test(masv88,locut(1),hicut(1),doref2(4),
     +                  midpt(1),1,c7)
         cmin3 = min(cmin3,c7)
         ngtests(3) = ngtests(3) + 1

       end if

c ... visible channel ratio test (channel 2 / channel 1)

       if (visusd) then

         if(snglnt) then
           ibin = int(pxrfa) + 1
           if(ibin .gt. 36) ibin = 36
           locut(1) = snglntvcl(ibin,1)
           locut(2) = snglntvcl(ibin,2)
           hicut(1) = snglntvch(ibin,1)
           hicut(2) = snglntvch(ibin,2)
           midpt(1) = snglntv(ibin,1)
           midpt(2) = snglntv(ibin,2)
         else
           locut(1) = dovratlo(1)
           locut(2) = dovrathi(1)
           hicut(1) = dovratlo(3)
           hicut(2) = dovrathi(3)
           midpt(1) = dovratlo(2)
           midpt(2) = dovrathi(2)
         end if
```

```
        vrat = masv88/masv66
        if (vrat.lt.midpt(1) .or. vrat.gt.midpt(2)) then
          nptests = nptests + 1
        end if
        call conf_test(vrat,locut,hicut,1.0,midpt,2,c8)
        cmin3 = min(cmin3,c8)
        ngtests(3) = ngtests(3) + 1

      end if

      if(nptests .eq. ngtests(3) .and. ngtests(3) .ne. 0) then
        call set_bit(testbits,12)
      end if

c ******       END OF GROUP 3 TESTS   *************************
c
c
c **********   START OF GROUP 4 TESTS  ***********************

      nptests = 0

c ... Near-infrared high cloud test.
      if (visusd .and. .not. snglnt) then
        if (masv188.le.doref3(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masv188,doref3(1),doref3(3),doref3(4),
     +                 doref3(2),1,c11)
        cmin4 = min(cmin4,c11)
        ngtests(4) = ngtests(4) + 1
      end if

      if(nptests .eq. ngtests(4) .and. ngtests(4) .ne. 0) then
        call set_bit(testbits,13)
      end if

c ***********   END OF GROUP 4 TESTS   *************************
c
c     Determine final confidence based on group values
      pre_confdnc = cmin1 * cmin2 * cmin3 * cmin4
      groups = 0.0
      do kk = 1,4
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

c     Visible thin cirrus check. This test has no effect on the
c     overall confidence of clear-sky.

c     The 1.88 reflectance should lie between the threshold and
c     the high confidence cutoff.
      if(masv188 .lt. doref3(2) .and. masv188 .ge. doref3(3)) then

c       Make sure that the 1.88 reflectance is not due to low-level
c       clouds which will also be bright in the visible.
        if(masv88 .lt. doref2(3)) then

          call clear_bit(testbits,8)

        end if
      end if

      return
      end
```

```
      subroutine ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     +                      confdnc)
c F77 ********************************************************
c
c       Routine for performing clear sky tests over water
c       surfaces during nightime hours.
c
c Input parameters:
c nbands         Total number of MAS channels.
c pxldat         Array containing reflectance or brightness temperatures
c                for all bands for a single pixel
c vza            Current pixel viewing angle
c pw             Amount of precipitable water at pixel site
c avgtherm       Average bt for a given box size
c!Output Parameters:
c testbits       two word 1-byte array containing bit results
c confdnc        product of all applied individual confidences
c
c END*************************************************************
c
      INCLUDE 'thresholds.inc'
c
c ... scalar arguments ..
      real confdnc,vza,pw
      real avgtherm(nbands)
      integer nbands
c ...
c ... array arguments ..
      real pxldat(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real c1,c2,c3,c4,ct1,ct2,diftsp1,diftsp2,dtr,mas11_4,masdf1,
     +     masdf2,masir11,masir12,masir13,masir4,pi,c5,pre_confdnc,
     +     groups,fac,masir8
      integer nptests
c ...
c ... local arrays ..
      real hicut(2),locut(2)
      integer ngtests(2)
c ...
c ... external functions ..
      real rega,regb
      external rega,regb
c ...
c ... external subroutines ..
      external conf_test,set_bit
c ...
c ... intrinsic functions ..
      intrinsic acos

c ... initialize variables
      pi = acos(-1.0)
      dtr = pi/180.0

c ... ngtests counts the number of tests applied within each test group
      ngtests(1) = 0
      ngtests(2) = 0

c ... Place band values into individual variables for easy
c ... identificatio.
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masir4 = pxldat(9)
      masir8 = pxldat(10)
      masir11 = avgtherm(11)
      masir12 = avgtherm(12)
      masir13 = avgtherm(13)

c ... the "cmin" variables represent group test confidences
      cmin1 = 1.0
      cmin2 = 1.0

C     **** GROUP 1 TESTS ***********************************

      nptests = 0

c     11 micron brigthness temperature threshold test.
      if (masir11 .ge. nobt11(2)) then
        nptests = nptests + 1
      end if
c ... calculate confidence compared to low and high confidence cutoffs
      call conf_test(masir11,nobt11(1),nobt11(3),nobt11(4),
     +               nobt11(2),1,c1)
      cmin1 = min(cmin1,c1)
      ngtests(1) = ngtests(1) + 1

c ... co2 high cloud test
      if (masir13.gt.noco2(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masir13,noco2(1),noco2(3),noco2(4),
     +               noco2(2),1,c2)
      cmin1 = min(cmin1,c2)
      ngtests(1) = ngtests(1) + 1

      if(nptests .eq. ngtests(1) .and. ngtests(1) .ne. 0) then
        call set_bit(testbits,10)
      end if

c     *****  END OF GROUP 1 TESTS  ************************
c
c     ****  GROUP 2 TESTS  *********************************

      nptests = 0

c ... Tri-spectral tests - apply only if a pw value exists for this
c ... pixel.
      if (pw .gt. 1.0) then

c ...   get dynamic clear sky thresholds based on pw
        diftsp2 = rega(pw)
        diftsp1 = regb(pw)
c ...   calculate 8 minus 11 and 11 minus 12 micron BTDIFs
        masdf2 = masir8 - masir11
        masdf1 = masir11 - masir12
        if ((masdf2.lt.diftsp2) .and. (masdf1.gt.diftsp1)) then
          nptests = nptests + 1
        end if
        locut(1) = diftsp2 + .5
        hicut(1) = diftsp2 - .5
        call conf_test(masdf2,locut,hicut,1.0,diftsp2,1,ct1)
        locut(1) = diftsp1 - .5
        hicut(1) = diftsp1 + .5
        call conf_test(masdf1,locut,hicut,1.0,diftsp1,1,ct2)
        c3 = ct1*ct2
        cmin2 = min(cmin2,c3)
        ngtests(2) = ngtests(2) + 1
        c4 = 0.0

      else

c ...   If no pw value exists for this pixel, use the low cloud
c ...   11-12 micron BTDIF test.

        masdf1 = masir11 - masir12
```

```
        if (masdf1.gt.no11_121o(2)) then
          nptests = nptests + 1
        end if
        call conf_test(masdf1,no11_121o(1),no11_121o(3),no11_121o(4),
     +                 no11_121o(2),1,c4)
        cmin2 = min(cmin2,c4)
        ngtests(2) = ngtests(2) + 1
        c3 = 0.0

      end if

c ... 11 minus 4 micron BTDIF fog and low cloud test.
      masl1_4 = masir11 - masir4
      if (masl1_4.le.no11_41o(2)) then
        nptests = nptests + 1
      end if
      call conf_test(masl1_4,no11_41o(1),no11_41o(3),no11_41o(4),
     +               no11_41o(2),1,c5)
      cmin2 = min(cmin2,c5)
      ngtests(2) = ngtests(2) + 1

      if(nptests .eq. ngtests(2) .and. ngtests(2) .ne. 0) then
        call set_bit(testbits,11)
      end if

c *******     END OF GROUP 2 TESTS *************************
c
c
c     Determine final confidence based on group values.
      pre_confdnc = cmin1 * cmin2
      groups = 0.0
      do kk = 1,2
        if(ngtests(kk) .gt. 0) then
          groups = groups + 1.0
        end if
      enddo
      fac = 1.0 / groups
      confdnc = pre_confdnc**fac

      return
      end
```

```
      subroutine polar_nite(pxldat,vza,eco_type,land,ice,snow,
     *                      uniform,pw,nlcntx,necntx,inband,nbands,
     *                      rgdata,avgtherm,diff,testbits,
     *                      confdnc)

c     Routine for providing conditional input parameters pertaining
c     to polar nighttime processing.

      integer*4 varslt
      real*4 pxldat(nbands),vza,confdnc,avgtherm(nbands),pw,
     *       rgdata(nlcntx,necntx,nbands),
     *       diff(nlcntx*necntx-1,nbands)
      logical*4 land,snow,ice,uniform
      byte eco_type,testbits(2)

      if(ice .or. snow) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,
     *                      avgtherm,testbits,confdnc)

      else if(land) then

        call LandNite_type1(nbands,pxldat,avgtherm,vza,eco_type,
     *                      testbits,confdnc)

      else

        call ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     *                  confdnc)

c       If confidence is still uncertain, apply the spatial variability
c       test. Also check for regional scene uniformity (see reg_anc.f).

        if(uniform .and. confdnc.le.0.95 .and. confdnc.gt.0.34)then

c         Get brightness temperature differences between pixel
c         of interest and the ones surrounding it.
          call get_regdif(nlcntx,necntx,inband,nbands,
     *                    rgdata,diff)

c         Check variation in the region.
          call spatial_var(nbands,nlcntx,necntx,diff,varslt)

c         Bump up the confidence if spatial variability test showed
c         uniform
          if ((varslt .eq. 1) .and. (confdnc .gt. .66)) then
            confdnc = 0.96
          else if ((varslt .eq. 1) .and. (confdnc .le. 0.66)) then
            confdnc = 0.67
          endif

        end if

      end if

      return
      end
```

```
      subroutine proc_path(water,land,coast,desert,day,snglnt,
     *                     snow,ice,testbits)

c F77 ************************************************************
c ... routine for determining processing path through
c ... the algorithm and setting the appropriate bit
c ... flags. This is where the appropriate day/night
c ... and land/water bits are set.
c
c Input parameters:
c water         Logical variable - true if water background
c land          Logical variable - true if land background
c coast         Logical variable - true if coastal background
c desert        Logical variable - true if desert background
c day           Logical variable - true if sza < 80
c snglnt        Logical variable - true if in sun-glint region
c snow          Logical variable - true if snow background
c ice           Logical variable - true if sea-ice background
c Output Parameters:
c testbits      two word 1-byte array containing bit results
c
c END************************************************************
c
c ... scalar arguments ..
      logical water,land,coast,desert,day,snglnt,snow,ice

c ... array arguments ..
      byte testbits(2)

c     Set snow/ice bit.
      if((.not. snow) .and. (.not. ice)) then
         call set_bit(testbits,4)
      end if

      if(.not. snglnt) then
c        Not in geometric sun-glint region. Set bit.
         call set_bit(testbits,3)
      end if

      if(day) then
c        Daytime.  Set bit.
         call set_bit(testbits,2)
      end if

c     Set coast, desert, or land processing path flags.  Default is
c     water (00), which is set during initialization of the bit flags.

      if(coast) then
c       Set coast bit.
        call set_bit(testbits,5)
      else if(desert) then
c       Set desert bit.
        call set_bit(testbits,6)
      else if(land) then
c        Set "land" bits.
         call set_bit(testbits,5)
         call set_bit(testbits,6)
      end if

      return
      end
```

```
      subroutine pxl_anc(nlcntx,necntx,rgvza,rgsza,rgrfa,rgpcwt,rgeco,
     *                   rglst,rglat,pxvza,pxrfa,pxpcwt,eco_type,
     *                   day,night,land,water,polar,snglnt,cnlwc,
     *                   visusd,vrused,ap_sea_ice,ap_snow,snow,ice,
     *                   coast,desert,nbands,pxldat,avgtherm,pxsza,jday)

c     Returns necessary ancillary data for a given pixel and
c     any logical flags needed for further processing.

      integer*4 rgpcwt(nlcntx,necntx),rglst(nlcntx,necntx),pxlst
      real*4 pxvza,pxrfa,pxpcwt,rgvza(nlcntx,necntx),
     *       rgrfa(nlcntx,necntx),rgsza(nlcntx,necntx),
     *       rglat(nlcntx,necntx),pxldat(nbands),
     *       avgtherm(nbands)
      logical*4 snglnt,visusd,water,land,day,night,polar,snow,
     *          ice,ap_sea_ice,ap_snow,dh_snow,desert,vrused,cnlwc,
     *          coast
      byte rgeco(nlcntx,necntx),eco_type

      ie = necntx - ((necntx-1) / 2)
      il = nlcntx - ((nlcntx-1) / 2)

      pxvza = rgvza(il,ie)
      pxsza = rgsza(il,ie)
      pxrfa = rgrfa(il,ie)
      pxlst = rglst(il,ie)
      pxpcwt = float(rgpcwt(il,ie))
      eco_type = rgeco(il,ie)
      rlat = rglat(il,ie)

c     Determine if current pixel is in geometric sun-glint region.
      if(pxrfa .le. 36.0) then
         snglnt = .true.
      else
         snglnt = .false.
      end if
c     Determine if current pixel will be processed as day or night.
      if(pxsza .lt. 85.0) then
         day = .true.
         visusd = .true.
         night = .false.
      else
         night = .true.
         day = .false.
         visusd = .false.
      end if

c     Determine if current pixel will be processed as land or water
c     (ocean).
      if(pxpcwt .ge. 90.0) then
         water = .true.
         land = .false.
      else
         land = .true.
         water = .false.
      end if

c     Determine if current pixel is in a polar region.
      if(rlat .gt. 60.0 .or. rlat .lt. -60.0) then
         polar = .true.
      else
         polar = .false.
      end if

c     Determine whether or not current pixel will be processed as
c     desert.
      if(eco_type .eq. 8 .or. eco_type .eq. 46 .or.
     *   eco_type .eq. 50 .or. eco_type .eq. 51 .or.
     *   eco_type .eq. 59 .or. eco_type .eq. 71 .or.
     *   eco_type .eq. 11) then
        desert = .true.
      else
        desert = .false.
      end if

c     Determine whether or not visible ratio test may be used over
c     land surfaces.
      if(eco_type .eq. 2 .or. eco_type .eq. 8 .or.
     *   eco_type .eq. 11 .or. eco_type .eq. 40 .or.
     *   eco_type .eq. 41 .or. eco_type .eq. 46 .or.
     *   eco_type .eq. 51 .or. eco_type .eq. 52 .or.
     *   eco_type .eq. 59 .or. eco_type .eq. 71 .or.
     *   eco_type .eq. 50) then
        vrused = .false.
      else
        vrused = .true.
      end if

c     Determine whether or not the current pixel will be processed as
c     coastal.
      if( (pxlst .eq. 2) .or. (.not. cnlwc) ) then
        coast = .true.
      else
        coast = .false.
      end if

c     In the future, we will have a priori information about surface
c     snow and sea ice cover. Set flags here, for now.
      ap_sea_ice = .false.
      ap_snow = .false.

c     Set flags indicating presence of surface snow and ice.

c     Initialize to "false".
      snow = .false.
      ice = .false.

      if(day) then
c       Run Bryan Baum's snow algorithm.
        call snowb(nbands,pxldat,rlat,pxsza,jday,avgtherm,dh_snow)
        if(land .and. dh_snow) snow = .true.
        if(water .and. dh_snow) ice = .true.
      else
c       Use a priori information.
        if(land .and. ap_snow) snow = .true.
        if(water .and. ap_sea_ice) ice = .true.
      end if

      return
      end
```

```
      integer function read_proc(tp_unit,eco_unit,maxlin,ibls,nelin,
     *                           ibes,nelet,inchns,islhms1,lst,pcwatr,
     *                           ecstypes,refang,rlats,rlons,sza,vza,raz,
     *                           indat,start,edge,gstart,gedge,eco_index,
     *                           dataid,latid,lonid,szaid,vzaid,solazid,
     *                           snsazid,sltid,block,count)

c-----------------------------------------------------------------------

      include 'cldmask.inc'

c-----------------------------------------------------------------------
c
c       external variables

      integer*4 tp_unit          ! unit No. of land/sea tag file
      integer*4 eco_unit         ! unit No. of ecosystem file
      integer*4 maxlin  ! number of lines in the image data file
      integer*4 ibls             ! start line No.
      integer*4 nelin            ! end line No.
      integer*4 ibes             ! start pixel No.
      integer*4 nelet            !,end pixel No.
      integer*4 inchns(inband,linmax)           ! input channel
      integer*4 islhms1          ! start time of the cloud mask (hhmmss)
      integer*4 lst(npixel,nlcntx,linmax)       ! land tag
      integer*4 pcwatr(npixel,nlcntx,linmax)    ! sea tag

      byte ecstypes(npixel,nlcntx,linmax)       ! ecosystem types

      real*4 refang(npixel,nlcntx,linmax)   ! solar reflectance angle
      real*4 rlats(npixel,nlcntx,linmax)    ! latitude
      real*4 rlons(npixel,nlcntx,linmax)    ! longitude
      real*4 sza(npixel,nlcntx,linmax)  ! solar zenith angle
      real*4 vza(npixel,nlcntx,linmax)  ! viewing zenith angle
      real*4 raz(npixel,nlcntx,linmax)  ! relative azimuth angle

      integer*2 indat(npixel,nbands,linmax)     ! image data

      integer*4 start(3)    ! start pixel/line to read from scanline
      integer*4 edge(3)     ! edge pixel/line to read from scanline
      integer*4 gstart(3)   ! start pixel/line to read from geometry
      integer*4 gedge(3)    ! edge pixel/line to read from geometry
      integer*4 eco_index   ! ecosystem file index (run parameter)
      integer*4 dataid      ! radiance data ID
      integer*4 lonid       ! longitude ID
      integer*4 szaid       ! solar zenith angle ID
      integer*4 vzaid       ! viewing zenith angle ID
      integer*4 solazid     ! solar azimuth ID
      integer*4 snsazid     ! sensor azimuth ID
      integer*4 sltid       ! scan line time ID
      integer*4 block       ! block = (nelin-ibls)/linmax
      integer*4 count       ! count[0,block]

c-----------------------------------------------------------------------
c
c       internal variables

      integer*4 lst_wk(npixel,nlcntx),pcwatr_wk(npixel,nlcntx)

      real*4
     *       rlat(npixel,linmax),rlon(npixel,linmax),
     *       rsza(npixel,linmax),
     *       rvza(npixel,linmax),rsolaz(npixel,linmax),
     *       rsnsaz(npixel,linmax)
      real*4 rlats_wk(npixel,nlcntx),rlons_wk(npixel,nlcntx),
     *       sza_wk(npixel,nlcntx),vza_wk(npixel,nlcntx),
     *       raz_wk(npixel,nlcntx),refang_wk(npixel,nlcntx)
      byte ecstypes_wk(npixel,nlcntx)

      integer*4 islhms,
     *          inchan(inband),
     *          stride(3),
     *          gstride(3),
     *          sfrdata,dhr2hms

      integer*4 line,nc,ich,irt,sltime,ifsc,il,ie

      integer*4 cnt,sline,eline
      integer*4 mlin /0/

c     For WINCE data.
      data inchan /2,3,7,9,10,15,20,30,31,42,45,46,49/

c     For SUCCESS data.
c     data inchan /1,2,7,9,10,15,20,31,32,42,45,46,49/

      data stride /1,1,1/
      data gstride /1,1,1/

c-----------------------------------------------------------------------

      include 'hdf.inc'

c-----------------------------------------------------------------------

c     Begin input scan line loop.

c     'mlin' counts # lines processed in current region (context).
c     'start(1)' is beginning pixel to read from scanline (always 0).
c     'edge(1)' is the # of pixels to read from scanline (always 'npixel').
c     'edge(2)' is the # of channels to read (always 1).
c     'edge(3)' is the # scanlines to read (always 1).
c     'stride(1,2,3)' is increment defined above (always 1).
c     'gstart(1)' is beginning pixel to read from geometry (always 0).
c     'gedge(1)' is the # of pixels to read from geometry (always 'npixel').
c     'gedge(2)' is the # geometry lines to read (always 1).
c     'gstride(1,2,3)' is increment defined above (always 1).
c     'line' is current line #.

c     write(*,'(1x,''# initialize start'')')
c     write(*,'(1x,''    HDF-file read '')')

      cnt = 0
      sline = ibls + linmax * count
      if(count.eq.block) then
         eline = nelin
      else
         eline = sline + linmax - 1
      endif
      write(*,*) 'count=',count,'    sline=',sline,'    eline=',eline

C      Do 100 line=ibls,nelin
      Do 100 line=sline,eline

         if(line .le. maxlin) then

c         write(*,'(1x,''# processing lines No.: '',i10)') line

c         Get data from a region encompassing the number of scans in a context
c         ('nlcntx') in the along-track direction and the total number of
c         pixels in the MAS scan (a "data block"). After the first data block
c         has been filled, only one additional scan is needed with each pass
c         through the loop.

          mlin = mlin + 1
```

```
read_proc.f          Wed Oct  [illegible]

          cnt = cnt + 1

c         Read scan line radiance data for input channels.
c         'start(3)' is the scan line number at which to begin.
c         'start(2)'is the channel number at which to begin.

c          write(*,*) 'cnt=', cnt
          start(3) = line - 1
c          write(*,*) 'start(3)=', start(3)
          do nc = 1,inband

            ich = inchan(nc)
            start(2) = ich - 1

c            write(*,*) 'nc=', nc

            irt = sfrdata(dataid,start,stride,edge,indat(1,nc,cnt))
c           write(*,'(1x,''data '',i5,i10)') ich,indat(1,nc,cnt)
            inchns(nc,cnt) = inchan(nc)

          enddo

c         Read latitudes, longitudes, solar zenith angles, viewing zenith
c         angles for current scan line.
c         'gstart(2)' is the scan line number at which to begin.

c         write(*,'(1x,''#     step.1'')')

          gstart(2) = line - 1
          irt = sfrdata(latid,gstart,gstride,gedge,rlat)
          irt = sfrdata(lonid,gstart,gstride,gedge,rlon)
          irt = sfrdata(szaid,gstart,gstride,gedge,rsza)
          irt = sfrdata(vzaid,gstart,gstride,gedge,rvza)
          irt = sfrdata(solazid,gstart,gstride,gedge,rsolaz)
          irt = sfrdata(snsazid,gstart,gstride,gedge,rsnsaz)

c         write(*,'(1x,''#     step.2'')')

c         Get scan line time.
          irt = sfrdata(sltid,line-1,stride,1,sltime)
          islhms = dhr2hms(sltime)
c         If middle scan of the first context, record the time. This is
c         the time of the first output record of the cloud mask.
          ifsc = (nlcntx-1) / 2
          if(line .eq. (ibls+ifsc)) islhmsl = dhr2hms(sltime)

c*****************************************************************

c         Calculate any additional geometry for necessary pixels on the current
c         write(*,'(1x,''#     step.3'')')
c         scan line.
          call get_geo(npixel,nlcntx,mdele1,mdele2,ibes,
     .                 nelet,mlin,rlat,rlon,rsza,rvza,rsolaz,rsnsaz,
     .                 rlats_wk,rlons_wk,sza_wk,vza_wk,
     .                 raz_wk,refang_wk)

C          write(6,*) 'rlat=',rlat(1,1),'     rlon=',rlon(1,1)
C          write(*,'(1x,''line, mlin'',i8,i8)') line, mlin
c         Get land/sea tag and ecosystem type for necessary pixels on the
c         current scan line.
          call get_sfc(npixel,nlcntx,ibes,nelet,mlin,rlats_wk,rlons_wk,
     .                 tp_unit,eco_index,eco_unit,ecstypes_wk,lst_wk,
     .                 pcwatr_wk)
c          write(*,'(1x,''#     step.4a'')')

          do 400 il = 1,nlcntx
            do 500 ie = ibes,nelet
              rlats(ie,il,cnt) = rlats_wk(ie,il)
              rlons(ie,il,cnt) = rlons_wk(ie,il)
              sza (ie,il,cnt) = sza_wk (ie,il)
              vza (ie,il,cnt) = vza_wk (ie,il)
              raz (ie,il,cnt) = raz_wk (ie,il)
              refang(ie,il,cnt) = refang_wk(ie,il)
              pcwatr(ie,il,cnt) = pcwatr_wk(ie,il)
              ecstypes(ie,il,cnt) = ecstypes_wk(ie,il)
              lst (ie,il,cnt) = lst_wk (ie,il)
500         continue
400       continue

          if(mlin .eq. nlcntx) then

            if(nlcntx .gt. 1) then

c             If not processing in single-line mode, re-buffer data in order
c             to add the next scan line and make a new data block.

              do 600 il = 2,nlcntx
                do 700 ie = ibes,nelet
                  rlats_wk(ie,il-1) = rlats_wk(ie,il)
                  rlons_wk(ie,il-1) = rlons_wk(ie,il)
                  sza_wk (ie,il-1) = sza_wk (ie,il)
                  vza_wk (ie,il-1) = vza_wk (ie,il)
                  raz_wk (ie,il-1) = raz_wk (ie,il)
                  refang_wk(ie,il-1) = refang_wk(ie,il)
                  pcwatr_wk(ie,il-1) = pcwatr_wk(ie,il)
                  ecstypes_wk(ie,il-1) = ecstypes_wk(ie,il)
                  lst_wk (ie,il-1) = lst_wk (ie,il)
700             continue
600           continue

              end if

              mlin = nlcntx - 1

            end if

          end if

100     continue

c           Normal return
c       write(*,'(1x,''# initialize end'')')
        return
        end
```

```
      subroutine reg_anc(npixel,nlcntx,necntx,jele,klin,
     *                   rlats,rlons,sza,vza,raz,refang,pcwatr,
     *                   ecstypes,rglat,rglon,rgsza,rgvza,rgraz,
     *                   rgrfa,rgpcwt,lst,rglst,rgeco,uniform,
     *                   cnlwc)

c     Routine which fills context (regional) ancillary data arrays.
c     Data is obtained from data block arrays.

      integer*4 pcwatr(npixel,nlcntx),ref_eco,lst(npixel,nlcntx),
     *          rglst(nlcntx,necntx),rgpcwt(nlcntx,necntx),pxpcwt

      real*4 sza(npixel,nlcntx),vza(npixel,nlcntx),
     *       raz(npixel,nlcntx),refang(npixel,nlcntx),
     *       rglat(nlcntx,necntx),rglon(nlcntx,necntx),
     *       rgsza(nlcntx,necntx),rgvza(nlcntx,necntx),
     *       rgraz(nlcntx,necntx),rgrfa(nlcntx,necntx),
     *       rlats(npixel,nlcntx),rlons(npixel,nlcntx)

      logical*4 uniform,ref_flag(4),cnlwc

      byte ecstypes(npixel,nlcntx),rgeco(nlcntx,necntx)

c     Fill regional context arrays.

      je = jele - 1
      kl = klin - 1
      do i = 1,nlcntx
        do j = 1,necntx
          rglat(i,j) = rlats(je+j,kl+i)
          rglon(i,j) = rlons(je+j,kl+i)
          rgsza(i,j) = sza(je+j,kl+i)
          rgvza(i,j) = vza(je+j,kl+i)
          rgraz(i,j) = raz(je+j,kl+i)
          rgrfa(i,j) = refang(je+j,kl+i)
          rgpcwt(i,j) = pcwatr(je+j,kl+i)
          rgeco(i,j) = ecstypes(je+j,kl+i)
          rglst(i,j) = lst(je+j,kl+i)
        enddo
      enddo

c     Check for regional uniformity. If any pixel in the current region
c     is inconsistent with the rest, then the region is non-uniform.
c     Categories are polar/non-polar, day/night, sun-glint, ecosystem,
c     and land/water.

      do i = 1,4
        ref_flag(i) = .true.
      enddo

      do i = 1,nlcntx
        do j = 1,necntx
          if(i .eq. 1 .and. j .eq. 1) then
            if(abs(rglat(i,j)) .le. 60.0) ref_flag(1) = .false.
            if(rgsza(i,j) .ge. 85.0) ref_flag(2) = .false.
            if(rgrfa(i,j) .gt. 36.0) ref_flag(3) = .false.
            if(rgpcwt(i,j) .ne. 100) ref_flag(4) = .false.
            ref_eco = rgeco(i,j)
          else
            if((abs(rglat(i,j)) .gt. 60.0) .neqv. ref_flag(1)) go to 100
            if((rgsza(i,j) .lt. 85.0) .neqv. ref_flag(2)) go to 100
            if((rgrfa(i,j) .le. 36.0) .neqv. ref_flag(3)) go to 100
            if((rgpcwt(i,j) .eq. 100) .neqv. ref_flag(4)) go to 100
            if(ref_eco .ne. rgeco(i,j)) go to 100
          end if
        enddo
      enddo

      uniform = .true.
      go to 200

  100 uniform = .false.

  200 continue

c     Determine if current pixel is in a coastal region.

      nland = 0
      nwater = 0
      ncoast = 0
      itotal = nlcntx * necntx

      do i = 1,nlcntx
        do j = 1,necntx

c         Check land/sea tag (% coverage of water in pixel).
          pxpcwt = rgpcwt(i,j)
          if(pxpcwt .eq. 100) then
            nwater = nwater + 1
          else if(pxpcwt .eq. 50) then
            ncoast = ncoast + 1
          else if(pxpcwt .eq. 0) then
            nland = nland + 1
          end if

        enddo
      enddo

      if( (nwater+ncoast) .eq. itotal) then
        cnlwc = .true.
      else if( (nland+ncoast) .eq. itotal) then
        cnlwc = .true.
      else
        cnlwc = .false.
      end if

      return
      end
```

```
      subroutine reg_data(npixel,nlcntx,necntx,jele,klin,
     *                    inband,nbands,rdat,rgdata)

c     Routine for getting regional reflectance and thermal
c     data from "data block" array.

      real*4 rdat(npixel,nlcntx,nbands),
     *       rgdata(nlcntx,necntx,nbands)

c     Fill regional context arrays.

      je = jele - 1
      kl = klin - 1
      do k = 1,inband
        do j = 1,nlcntx
          do i = 1,necntx
            rgdata(j,i,k) = rdat(je+i,kl+j,k)
          enddo
        enddo
      enddo

      return
      end
```

```
      real function rega(pw)

c F77 ********************************************************
c ...
c ... regression function relating the pw values to the 8-11um btdif.  the
c ... difference is related to the amount of pw in the atmosphere
c ... due to the weak water vapor lines present in this spectral
c ... region.  the regressions were determined from actual hirs
c ... data, and tweaked for use with the mas bandwidths.  this
c ... information provides a threshold value for 8-11um clear sky
c ... determination.
c ...
c Input parameters:
c pw - total column precipitable water value (g/cm-2)
c Output Parameters:
c rega - 8-11um clear sky threshold
c
c END*************************************************************
c
c ...
c ... scalar arguments ..
      real pw
c ...
c ... local scalars ..
c ... variables: a - slope of regression
c ... b - intercept of regression
      real a,b
c ...
c ... intrinsic functions ..
      intrinsic log
c ...
c ... data statements ..
c ... coefficients for mas data
      data a/-1.64805/,b/-3.19767/
c ...
c ...
      rega = (a*log(pw)) + b
c ...
      return
      end
```

```
      real function regb(pw)

c P77 ************************************************************
c ...
c ... regression function relating the pw values to the 11-12um btdif.  the
c ... difference is related to the amount of pw in the atmosphere
c ... due to the weak water vapor lines present in this spectral
c ... region.  the regressions were determined from actual hirs
c ... data, and tweaked for use with the mas bandwidths.  this
c ... information provides a threshold value for 11-12um clear sky
c ... determination.
c ...
c Input parameters:
c pw - total column precipitable water value (g/cm-2)
c Output Parameters:
c regb - 8-11um clear sky threshold
c
c END*************************************************************

c ...
c ... scalar arguments ..
      real pw
c ...
c ... local scalars ..
c ... variables: c - slope of regression
c ... d - intercept of regression
      real c,d
c ...
c ... data statements ..
c ... coefficients for mas data
      data c/0.488198/,d/-0.456924/
c ...
c ...
      regb = (c*pw) + d
c ...
      return
      end
```

```
      subroutine set_HDF(if_hdfid,dataid,latid,lonid,szaid,vzaid,
     *                   slcid,sltid,slymdid,cwlid,solirid,solazid,
     *                   snsazid)

c     Routine which retrieves HDF indices for reading the input HDF file.

      integer*4 dataid,latid,lonid,szaid,vzaid,slcid,sltid,slymdid,
     *          cwlid,solirid,solazid,snsazid,sds_indx,sfn2index,
     *          sfselect

      include 'hdf.inc'

      sds_indx = sfn2index(if_hdfid,'CalibratedData')
      dataid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'Central100%ResponseWavelength')
      cwlid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'PixelLongitude')
      lonid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'PixelLatitude')
      latid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SolarZenithAngle')
      szaid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SensorZenithAngle')
      vzaid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'ScanLineCounter')
      slcid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'ScanlineTime')
      sltid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'YearMonthDay')
      slymdid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SolarSpectralIrradiance')
      solirid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SolarAzimuthAngle')
      solazid = sfselect(if_hdfid,sds_indx)

      sds_indx = sfn2index(if_hdfid,'SensorAzimuthAngle')
      snsazid = sfselect(if_hdfid,sds_indx)

      return
      end
```

```
      subroutine set_bit(testbits,bit_num)

c     Routine for setting a single bit within the 48-bit
c     cloud mask output array.

      integer*4 bit_num
      byte testbits(2)

c     Determine which word (1-2) of the 1-byte array contains the
c     bit of interest.

      iword = (bit_num / 8) + 1

c     Determine the position of the bit within the current
c     8-bit segment (1-byte word).

      ipos = bit_num - ((iword-1) * 8)

      itest = testbits(iword)
      itest = ibset(itest,ipos)

      testbits(iword) = itest

c     write(*,'(1x,''set '',4i10)') iword,ipos,itest,testbits(iword)

      return
      end
```

```
      subroutine set_confdnc(confdnc,testbits)

c     Routine for setting output "bit" flags according
c     to final confidence of clear sky.

      byte testbits(2)

      if(confdnc .gt. 0.99) then
        call set_bit(testbits,0)
        call set_bit(testbits,1)
      else if(confdnc .gt. 0.95) then
        call set_bit(testbits,1)
      else if(confdnc .gt. 0.66) then
        call set_bit(testbits,0)
      end if

      return
      end
```

```
      subroutine shadows(nbands,pxldat,shadow,testbits)

c F77 ********************************************************
c ... determines the presence of shadows and sets the
c ... appropriate bit flag.
c
c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c Output Parameters:
c shadow        logical variable indicating shadow is present if .true.
c testbits      two word 1-byte array containing test results
c
c END*********************************************************
c
      include 'thresholds.inc'

c ... scalar arguments ..
      integer nbands
      logical*4 shadow
c ...
c ... array arguments ..
      real pxldat(nbands)
      byte testbits(2)
c ...
c ... local scalars ..
      real masv88,masv66,masv945,masv162
c ...
      masv66 = pxldat(2)
      masv88 = pxldat(3)
      masv945 = pxldat(4)

c     Reflectance at 0.945 um must be ge 12% and "visible ratio" gt the
c     ocean threshold or else we're seeing a shadow.  The test on the
c     ratio is to assure that the scene is not clear-sky conditions over
c     a sub-grid scale water body.

      vrat = masv88 / masv66

      if(masv945 .lt. 12.0 .and. vrat .gt. dovratlo(2)) then
        shadow = .true.
        call clear_bit(testbits,9)
      else
        shadow = .false.
      end if

      return
      end
```

```
      subroutine snowb(nbands,pxldat,rlat,sza,jday,avgtherm,
     *                 dh_snow)
c
c F77 *********************************************************
c
c Bryan Baum's snow test.

c Input parameters:
c nbands        Total number of MAS channels.
c pxldat        Array containing reflectance or brightness temperatures
c               for all bands for a single pixel
c rlat          latitude
c sza           solar zenith angle
c avgtherm      Array containing regional means of IR brightness temps.
c jday          Day of year
c Output Parameters:
c dh_snow       logical variable indicating the prescence of snow or
c               ice in a FOV.
c
c END*************************************************************
c
      include 'thresholds.inc'   '
c
c ... parameters ..
c ...
c ... scalar arguments ..
      logical dh_snow
c ...
c ... array arguments ..
      integer jday
      real pxldat(nbands),avgtherm(nbands),rlat,sza
c ...
c ... local scalars ..
      real masir11,masv66,masir37,ref3,ref3o1,irdif
c
c ... Some tests may use a combination of single-pixel and
c ... averaged values.
      masv66 = pxldat(2)
      masir37 = avgtherm(8)
      masir11 = avgtherm(11)
      call get_ref13(masir37,masir11,jday,sza,ref3)
      ref3o1 = ref3 / masv66
      irdif = masir37 - masir11
c
      if(abs(rlat) .gt. 50.0 .and. masir11 .le. 260.0) then

        if(ref3o1 .le. .06 .or. (masv66 .ge. 25.0 .and.
     *     ref3 .le. 5.0 .and. irdif .le. 16.0)) then

          dh_snow = .true.

        else

          dh_snow = .false.

        end if

      else

        if(masv66 .ge. 20.0 .and. masir11 .le. 277.0 .and.
     *     ref3 .le. 3.0 .and. irdif .le. 8.0) then

          dh_snow = .true.

        else

          dh_snow = .false.

        end if

      end if

      return
      end
```

```
      REAL FUNCTION SOLR_IRRAD(WAVLEN,IDAY)

C     PURPOSE          CALCULATE SOLAR SPECTRAL IRRADIANCE INCIDENT AT
C                      TOP OF EARTH'S ATMOSPHERE AT A GIVEN WAVELENGTH.
C                      ALLOWANCE IS MADE FOR THE ELLIPTICAL ORBIT
C                      OF THE EARTH ABOUT THE SUN.
C                      BASED ON LOWTRAN7 SUBROUTINE "SOURCE"
C     INPUTS
C     WAVLEN           DESIRED WAVELENGTH IN MICRONS
C     DAY              DAY OF YEAR
C     OUTPUT
C     SOLAR            SOLAR SPECTRAL IRRADIANCE (MILLIWATTS/CM**2/MICR
C-----------------------------------------------------------------------

      DIMENSION NDAY(13),RAT(13)
      DATA NDAY/1,32,60,91,121,152,182,213,244,274,305,335,366/
      DATA RAT/1.034,1.030,1.019,1.001,.985,.972,.967,.971,.982,
     1    .998,1.015,1.029,1.034/
CHANGE*** 2 AUG 89 LG
c     WRITE(*,*)'ENTERING SOLAR ',wavlen, iday
      VV=10000.0/WAVLEN
      V=10000./VV
      IF(VV.LE.0.) V = 1.0E+38
C
C     SUN ELLIPTIC ORBIT FACTOR
C
      FORBIT=0.0
      IF(IDAY.GT.0 .AND. IDAY.LT.367) GO TO 55
      FORBIT = 1.0
      GO TO 90
55    CONTINUE
      DO 60 I=1,13
      IF(NDAY(I).EQ.IDAY) GO TO 80
      IF(NDAY(I).GT.IDAY) GO TO 70
60    CONTINUE
70    FORBIT=RAT(I-1)+(IDAY-NDAY(I-1))*(RAT(I)-RAT(I-1))/(NDAY(I)
     1   -NDAY(I-1))
      GO TO 90
80    FORBIT=RAT(I)
90    CONTINUE
C
C     SOLAR INTENSITY
C
      SS = SUN(VV)*FORBIT
C     CONVERT W/M-2-MICRON TO MW/CM-2-MICRON
C     SOLR_IRRAD = SS*.1
      SOLR_IRRAD = SS
C
      RETURN
      END
C
      FUNCTION   SUN(V)
C
C        EVALUATES THE EXTRA-TERRESTRIAL SOLAR IRRADIANCE
C
C        INPUT:  V  =  FREQUENCY    (CM-1)
C                VALID RANGE    0 TO 57490 (CM-1)
C                    (EQUIVALENT TO WAVELENGTHS > 0.174 MICROMETERS)
C
C        OUTPUT:  SUN  =  SOLAR IRRADIANCE  (WATTS M-2 MICROMETER-1)
C
C        WRITES A WARNING MESSAGE TO TAPE6  &  RETURNS  SUN = 0
C            IF THE INPUT FREQUENCY IS OUT OF RANGE
C
C      USES  BLOCK DATA SOLAR   WHICH CONTAINS THE VALUES FOR SOLARA +
C
      COMMON /SUNDAT/ SOLARA(1440), SOLARB(2910)
      DATA  A, B / 3.50187E-13, 3.93281 /
C
C       WM, W0, W1, W2  ARE STATEMENT FUNKTIONS USED BY
C            THE 4 POINT LAGRANGE INTERPOLATION
      WM(P) = P*(P - 1)*(P - 2)
      W0(P) = 3*(P**2 - 1)*(P - 2)
      W1(P) = 3*P*(P + 1)*(P - 2)
      W2(P) = P*(P**2 - 1)
C
C           IF  V  IS TOO SMALL,  WRITE WARNING  +  RETURN SUN = 0
      IF(V .LT. 0.0) THEN
        SUN = 0.0
C       WRITE(6, 900)'MAS WAVENUMBER LESS THAN ZERO ',V
        RETURN
C
      ELSEIF( V .GE. 0.0  .AND.  V .LT. 100.0 ) THEN
C         FOR LOW FREQUENCIES USE A POWER LAW APPROXIMATION
        SUN = A*V**B
        RETURN
C
      ELSEIF( V .GE. 100.0  .AND.  V .LT. 28420.0 ) THEN
C         USE  4 POINT INTERPOLATION  ON  ARRAY  SOLARA
C              WHICH IS AT  20 CM-1  SPACING  FROM 0 TO 28720 CM-1
        I = 1 + INT(V/20.0)
        P = MOD(V, 20.0)/20.0
        SUN = ( W2(P)*SOLARA(I+2) - W1(P)*SOLARA(I+1) +
     +              W0(P)*SOLARA(I) - WM(P)*SOLARA(I-1) ) / 6
        RETURN
C
      ELSEIF( V .GE. 28420.0  .AND.  V .LE. 57470.0 ) THEN
C         USE  4 POINT INTERPOLATION  ON  ARRAY  SOLARB
C             WHICH IS AT  10 CM-1  SPACING  FROM 28400 TO 57490 CM-1
        I = INT(V/10.0) - 2839
        P = MOD(V, 10.0)/10.0
        SUN = ( W2(P)*SOLARB(I+2) - W1(P)*SOLARB(I+1) +
     +              W0(P)*SOLARB(I) - WM(P)*SOLARB(I-1) ) / 6
        RETURN
C
      ELSEIF( V .GT. 57470.0 ) THEN
C           IF  V  IS TOO LARGE,  WRITE WARNING  +  RETURN SUN = 0
        SUN = 0.0
        WRITE(6, 900)'MAS WAVENUMBER TOO LARGE ',V
        RETURN
C
      ENDIF
C
      RETURN
  900 FORMAT('0 *****  WARNING - INPUT FREQUENCY = ', 1PG12.5, 'CM-1',
     +  /, '   OUTSIDE VALID RANGE OF 0 TO 57470 CM-1    *******', / )
      END
      BLOCK DATA SOLARS
C>    BLOCK DATA
C
C
C     COMMON /SUNDAT/   SOLARA(1440), SOLARB(2910)
      COMMON /SUNDAT/   SUNA01( 41),SUNA02(144),SUNA03(144),SUNA04(144),
     A      SUNA05(144),SUNA06(144),SUNA07(144),SUNA08(144),SUNA09(144),
     B      SUNA10(144),SUNA11(103),SUNB01(144),SUNB02(144),SUNB03(144),
     C      SUNB04(144),SUNB05(144),SUNB06(144),SUNB07(144),SUNB08(144),
     D      SUNB09(144),SUNB10(144),SUNB11(144),SUNB12(144),SUNB13(144),
     E      SUNB14(144),SUNB15(144),SUNB16(144),SUNB17(144),SUNB18(144),
     F      SUNB19(144),SUNB20(144),SUNB21( 30)
C
C          SOLAR SPECTRUM FROM      0 TO    800 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA01  /
     A 0.0000E+00, 4.5756E-08, 7.0100E-07, 3.4580E-06, 1.0728E-05,
     B 2.5700E-05, 5.2496E-05, 9.6003E-05, 1.6193E-04, 2.5766E-04,
     C 3.9100E-04, 5.6923E-04, 8.0203E-04, 1.1006E-03, 1.4768E-03,
     D 1.9460E-03, 2.5213E-03, 3.2155E-03, 4.0438E-03, 5.0229E-03,
```

```
     E 6.1700E-03, 7.5145E-03, 9.0684E-03, 1.0853E-02, 1.2889E-02,
     F 1.5213E-02, 1.7762E-02, 2.0636E-02, 2.3888E-02, 2.7524E-02,
     G 3.1539E-02, 3.5963E-02, 4.0852E-02, 4.6236E-02, 5.2126E-02,
     H 5.8537E-02, 6.5490E-02, 7.3017E-02, 8.1169E-02, 9.0001E-02,
     I 9.9540E-02 /
C         SOLAR SPECTRUM FROM    820 TO   3680 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA02  /
     A .10980, .12080, .13260, .14520, .15860, .17310, .18850, .20490,
     B .22240, .24110, .26090, .28200, .30430, .32790, .35270, .37890,
     C .40650, .43550, .46600, .49800, .53160, .56690, .60390, .64260,
     D .68320, .72560, .76990, .81620, .86440, .91470, .96710, 1.0220,
     E 1.0780, 1.1370, 1.1990, 1.2630, 1.3290, 1.3990, 1.4710, 1.5460,
     F 1.6250, 1.7060, 1.7910, 1.8800, 1.9710, 2.0670, 2.1660, 2.2680,
     G 2.3740, 2.4840, 2.5970, 2.7140, 2.8350, 2.9600, 3.0890, 3.2210,
     H 3.3570, 3.4980, 3.6420, 3.7900, 3.9440, 4.1040, 4.2730, 4.4450,
     I 4.6150, 4.7910, 4.9830, 5.1950, 5.4210, 5.6560, 5.8930, 6.1270,
     J 6.3560, 6.5820, 6.8080, 7.0360, 7.2700, 7.5170, 7.7890, 8.0910,
     K 8.4070, 8.7120, 8.9900, 9.2490, 9.5000, 9.7550, 10.010, 10.250,
     L 10.480, 10.700, 10.950, 11.230, 11.550, 11.900, 12.250, 12.600,
     M 12.930, 13.250, 13.530, 13.780, 14.040, 14.320, 14.660, 15.070,
     N 15.530, 16.011, 16.433, 16.771, 17.077, 17.473, 17.964, 18.428,
     O 18.726, 18.906, 19.141, 19.485, 19.837, 20.160, 20.509, 21.024,
     P 21.766, 22.568, 23.190, 23.577, 23.904, 24.335, 24.826, 25.236,
     Q 25.650, 26.312, 27.208, 27.980, 28.418, 28.818, 29.565, 30.533,
     R 31.247, 31.667, 32.221, 33.089, 33.975, 34.597, 35.004, 35.395 /
C         SOLAR SPECTRUM FROM   3700 TO   6560 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA03  /
     A 36.026, 36.985, 37.890, 38.401, 38.894, 39.857, 40.926, 41.570,
     B 42.135, 43.083, 44.352, 45.520, 45.982, 46.281, 48.335, 51.987,
     C 54.367, 54.076, 52.174, 50.708, 52.153, 55.707, 56.549, 54.406,
     D 53.267, 56.084, 61.974, 64.406, 60.648, 55.146, 53.067, 57.476,
     E 64.645, 68.348, 69.055, 69.869, 70.943, 71.662, 72.769, 74.326,
     F 75.257, 74.883, 73.610, 73.210, 74.886, 78.042, 80.204, 80.876,
     G 82.668, 84.978, 86.244, 88.361, 91.998, 95.383, 98.121, 100.29,
     H 100.64, 99.997, 101.82, 105.06, 107.50, 109.99, 112.45, 113.90,
     I 113.79, 119.23, 121.96, 124.58, 127.14, 125.19, 124.37, 125.00,
     J 127.88, 130.67, 131.98, 133.74, 136.69, 136.18, 135.02, 137.44,
     K 138.44, 137.25, 136.35, 142.60, 144.54, 148.37, 151.90, 151.55,
     L 155.35, 157.59, 159.70, 162.28, 168.44, 171.43, 169.82, 170.33,
     M 172.28, 176.68, 181.92, 186.06, 187.85, 186.00, 189.82, 189.35,
     N 192.86, 202.00, 209.63, 205.76, 212.88, 215.63, 216.51, 219.20,
     O 220.29, 221.12, 227.12, 229.97, 233.23, 233.95, 234.52, 234.45,
     P 235.77, 239.80, 243.11, 241.19, 242.34, 243.69, 242.84, 246.19,
     Q 246.11, 246.76, 251.75, 255.38, 258.74, 260.26, 263.40, 268.68,
     R 271.81, 272.95, 273.93, 274.74, 274.43, 279.69, 287.76, 287.72 /
C         SOLAR SPECTRUM FROM   6580 TO   9440 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA04  /
     A 287.96, 290.01, 291.92, 295.28, 296.78, 300.46, 302.19, 299.14,
     B 301.43, 305.68, 309.29, 310.63, 313.24, 314.61, 309.58, 318.81,
     C 320.54, 321.62, 328.58, 331.66, 337.20, 345.62, 345.54, 342.96,
     D 344.38, 346.23, 349.17, 351.79, 354.71, 356.97, 358.29, 362.29,
     E 364.15, 364.97, 367.81, 368.98, 369.07, 372.17, 377.79, 381.25,
     F 384.22, 388.66, 393.58, 396.98, 398.72, 400.61, 404.06, 408.23,
     G 412.47, 415.58, 416.17, 416.53, 419.55, 425.88, 433.30, 437.73,
     H 438.13, 439.79, 441.51, 438.71, 434.25, 437.54, 448.95, 448.86,
     I 439.46, 437.10, 439.34, 444.33, 455.00, 467.05, 473.04, 469.64,
     J 467.53, 473.78, 477.50, 477.50, 480.96, 483.94, 482.19, 479.08,
     K 482.09, 493.43, 498.40, 492.05, 489.53, 493.34, 495.51, 496.52,
     L 499.57, 504.65, 509.68, 512.00, 512.05, 512.31, 515.00, 520.70,
     M 527.30, 531.88, 532.16, 530.48, 532.33, 539.26, 548.57, 553.00,
     N 548.96, 546.05, 551.00, 556.41, 557.21, 557.85, 560.95, 564.02,
     O 565.57, 566.38, 567.88, 571.48, 576.68, 581.54, 586.51, 593.62,
     P 600.70, 602.79, 601.39, 603.00, 606.88, 605.95, 600.97, 600.79,
     Q 607.21, 612.87, 614.13, 614.39, 616.61, 620.53, 625.19, 629.78,
     R 633.79, 637.31, 640.47, 642.53, 642.62, 641.93, 643.11, 646.68 /
C         SOLAR SPECTRUM FROM   9460 TO  12320 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA05  /
     A 650.57, 654.30, 660.95, 672.10, 682.31, 684.89, 682.20, 682.53,
     B 687.79, 691.42, 689.62, 688.14, 693.71, 703.25, 708.07, 706.22,
     C 704.64, 708.97, 717.35, 725.43, 731.08, 734.17, 735.41, 736.60,
     D 739.34, 742.90, 745.04, 744.29, 742.44, 749.53, 755.70, 758.82,
     E 766.31, 761.53, 762.09, 769.68, 764.18, 763.75, 768.88, 762.69,
     F 753.93, 762.38, 765.79, 772.19, 760.67, 762.10, 766.76, 766.98,
     G 769.35, 773.50, 766.84, 763.60, 773.82, 777.18, 779.61, 792.48,
     H 797.54, 787.81, 793.75, 805.96, 804.77, 806.62, 821.72, 830.28,
     I 827.54, 831.06, 830.20, 826.22, 823.28, 822.18, 833.92, 854.58,
     J 859.80, 862.56, 871.16, 875.16, 867.67, 863.87, 883.30, 893.40,
     K 897.74, 905.24, 905.38, 911.07, 930.21, 939.24, 934.74, 935.15,
     L 942.38, 948.13, 947.00, 951.88, 960.12, 951.88, 954.22, 959.07,
     M 963.36, 980.16, 983.66, 978.76, 979.38, 985.24, 977.08, 919.94,
     N 899.68, 962.91, 997.17, 999.93, 995.65, 999.93, 1014.9, 951.57,
     O 893.52, 955.14, 1003.1, 990.13, 978.79, 1011.2, 1034.7, 1031.9,
     P 1029.9, 1039.7, 1045.5, 1044.1, 1049.6, 1056.1, 1049.8, 1038.0,
     Q 1051.9, 1072.2, 1075.5, 1077.0, 1079.3, 1078.0, 1075.7, 1079.7,
     R 1081.0, 1069.8, 1078.4, 1104.3, 1111.4, 1111.7, 1117.6, 1119.6 /
C         SOLAR SPECTRUM FROM  12340 TO  15200 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA06  /
     A 1109.3, 1100.6, 1112.9, 1122.7, 1119.5, 1123.9, 1136.1, 1143.7,
     B 1140.5, 1141.2, 1151.5, 1148.7, 1138.3, 1141.0, 1150.6, 1160.1,
     C 1170.6, 1177.7, 1179.8, 1181.7, 1182.4, 1179.8, 1181.8, 1188.3,
     D 1190.0, 1191.4, 1197.0, 1196.0, 1192.2, 1200.6, 1210.4, 1209.1,
     E 1207.5, 1205.3, 1193.3, 1192.9, 1220.0, 1243.3, 1245.4, 1241.5,
     F 1240.2, 1241.1, 1244.0, 1248.5, 1253.2, 1257.1, 1259.9, 1261.9,
     G 1263.6, 1265.7, 1269.6, 1277.0, 1284.2, 1284.4, 1282.7, 1287.2,
     H 1286.8, 1272.3, 1262.2, 1270.7, 1288.8, 1304.8, 1311.8, 1312.2,
     I 1314.4, 1320.2, 1326.2, 1328.4, 1325.3, 1322.5, 1325.4, 1334.6,
     J 1346.4, 1354.0, 1353.7, 1347.3, 1338.3, 1331.0, 1329.7, 1338.0,
     K 1351.9, 1363.0, 1368.8, 1372.0, 1375.9, 1382.1, 1387.8, 1388.8,
     L 1388.2, 1392.2, 1401.7, 1412.9, 1418.2, 1410.7, 1395.9, 1385.7,
     M 1388.1, 1405.0, 1424.0, 1428.1, 1422.2, 1423.6, 1434.5, 1445.2,
     N 1450.7, 1451.8, 1451.5, 1453.9, 1459.9, 1466.9, 1471.3, 1469.4,
     O 1462.5, 1460.4, 1468.9, 1481.8, 1490.8, 1495.3, 1497.9, 1500.7,
     P 1505.2, 1510.0, 1512.3, 1512.7, 1515.6, 1521.6, 1524.2, 1520.7,
     Q 1520.3, 1531.6, 1545.7, 1548.2, 1541.7, 1542.2, 1553.6, 1563.6,
     R 1563.6, 1559.9, 1561.3, 1569.9, 1581.6, 1577.6, 1529.7, 1447.0 /
C         SOLAR SPECTRUM FROM  15220 TO  18080 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA07  /
     A 1396.9, 1428.7, 1506.4, 1567.1, 1594.0, 1606.1, 1613.5, 1609.0,
     B 1588.6, 1567.8, 1567.3, 1587.2, 1610.2, 1624.4, 1630.2, 1630.9,
     C 1628.1, 1622.3, 1616.9, 1618.9, 1631.6, 1648.1, 1658.2, 1659.7,
     D 1658.1, 1658.0, 1659.4, 1660.4, 1659.2, 1653.7, 1645.3, 1642.1,
     E 1652.7, 1674.2, 1694.1, 1700.6, 1703.4, 1697.6, 1654.5, 1644.4,
     F 1661.6, 1676.3, 1707.7, 1703.1, 1710.8, 1732.3, 1716.5, 1719.6,
     G 1729.6, 1683.1, 1628.5, 1683.5, 1727.0, 1707.8, 1689.4, 1698.4,
     H 1733.1, 1737.8, 1714.1, 1734.6, 1750.1, 1750.1, 1760.3, 1764.3,
     I 1765.3, 1769.4, 1779.9, 1793.0, 1765.1, 1729.4, 1745.9, 1753.4,
     J 1758.1, 1775.0, 1768.4, 1767.9, 1789.5, 1806.6, 1799.3, 1782.6,
     K 1779.3, 1792.1, 1809.7, 1808.0, 1794.4, 1818.6, 1774.2, 1648.5,
     L 1674.3, 1789.3, 1847.2, 1848.3, 1812.9, 1796.4, 1840.3, 1868.3,
     M 1864.6, 1873.2, 1872.2, 1856.0, 1845.0, 1842.4, 1823.9, 1795.1,
     N 1819.6, 1861.5, 1857.7, 1838.6, 1840.5, 1863.5, 1876.8, 1884.4,
     O 1894.9, 1875.2, 1821.2, 1779.4, 1810.2, 1855.3, 1831.8, 1837.3,
     P 1882.3, 1866.4, 1819.6, 1804.8, 1831.4, 1861.6, 1867.1, 1862.9,
     Q 1851.9, 1834.7, 1835.2, 1845.1, 1831.9, 1803.6, 1792.5, 1821.8,
     R 1845.8, 1832.3, 1847.6, 1894.2, 1909.2, 1901.0, 1891.2, 1869.9 /
C         SOLAR SPECTRUM FROM  18100 TO  20960 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA08  /
     A 1854.4, 1865.8, 1873.7, 1868.8, 1881.7, 1897.1, 1884.2, 1856.2,
     B 1840.6, 1855.1, 1885.3, 1903.6, 1900.1, 1887.4, 1887.7, 1879.0,
     C 1844.5, 1844.1, 1877.1, 1847.3, 1785.1, 1792.6, 1848.7, 1894.4,
     D 1908.8, 1892.8, 1867.4, 1885.6, 1959.9, 1971.9, 1895.8, 1883.5,
     E 1917.6, 1853.8, 1793.0, 1875.6, 1974.0, 1975.7, 1943.9, 1926.4,
     F 1914.4, 1902.7, 1882.5, 1813.3, 1710.8, 1717.9, 1859.7, 1965.1,
     G 1970.1, 1941.4, 1902.5, 1852.0, 1836.3, 1879.3, 1901.6, 1862.9,
     H 1839.1, 1840.9, 1780.0, 1684.9, 1677.3, 1718.7, 1697.3, 1684.3,
     I 1784.5, 1898.0, 1910.3, 1877.2, 1866.6, 1862.6, 1860.3, 1899.7,
```

```
     J 1971.0, 1999.9, 1970.9, 1936.5, 1922.8, 1922.8, 1924.0, 1917.2,
     K 1912.0, 1926.2, 1959.7, 1995.4, 1995.9, 1938.8, 1883.5, 1894.7,
     L 1933.3, 1935.1, 1899.3, 1852.7, 1820.2, 1821.5, 1865.2, 1935.5,
     M 1966.1, 1919.6, 1881.2, 1931.5, 2015.6, 2050.0, 2021.4, 1960.8,
     N 1938.2, 1997.0, 2051.0, 2003.4, 1912.1, 1880.2, 1895.2, 1898.0,
     O 1898.8, 1938.3, 1994.2, 2010.0, 1982.4, 1948.8, 1927.3, 1911.6,
     P 1877.7, 1791.6, 1679.8, 1645.0, 1727.3, 1845.2, 1926.2, 1973.4,
     Q 2005.2, 2021.6, 2021.8, 2025.7, 2054.3, 2086.5, 2082.6, 2052.9,
     R 2047.1, 2070.2, 2072.4, 2038.1, 2020.2, 2049.9, 2074.0, 2038.1 /
C        SOLAR SPECTRUM FROM  20980 TO  23840 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA09  /
     A 1978.6, 1963.5, 1996.8, 2037.5, 2057.5, 2048.2, 2018.4, 1999.2,
     B 2011.4, 2039.5, 2055.0, 2040.2, 1981.8, 1911.4, 1891.8, 1938.3,
     C 1991.7, 2005.5, 2000.8, 2011.3, 2022.7, 1997.5, 1947.7, 1936.3,
     D 1986.6, 2037.9, 2032.8, 1995.7, 1984.0, 2012.0, 2055.5, 2091.6,
     E 2106.5, 2094.9, 2070.4, 2052.8, 2046.7, 2043.8, 2035.5, 2016.6,
     F 1988.4, 1973.3, 1999.0, 2057.4, 2103.8, 2109.4, 2089.4, 2068.5,
     G 2051.8, 2031.2, 2005.9, 1986.7, 1981.5, 1979.4, 1964.1, 1943.6,
     H 1951.8, 2007.3, 2083.2, 2139.1, 2158.0, 2143.3, 2103.2, 2050.9,
     I 2001.9, 1974.5, 1988.0, 2037.8, 2075.1, 2050.6, 1971.5, 1884.5,
     J 1828.5, 1820.9, 1866.4, 1935.3, 1974.2, 1958.7, 1925.1, 1920.2,
     K 1949.7, 1984.6, 1996.4, 1966.4, 1884.8, 1781.9, 1726.8, 1759.4,
     L 1817.4, 1800.4, 1692.6, 1593.2, 1598.6, 1700.3, 1823.8, 1909.7,
     M 1937.7, 1902.5, 1822.4, 1737.8, 1683.2, 1666.8, 1682.7, 1715.3,
     N 1734.1, 1712.4, 1668.2, 1655.0, 1698.1, 1727.2, 1636.9, 1415.7,
     O 1204.2, 1155.8, 1278.4, 1450.0, 1560.5, 1595.1, 1587.8, 1570.6,
     P 1565.8, 1590.3, 1640.5, 1688.4, 1708.1, 1703.6, 1700.7, 1718.5,
     Q 1749.0, 1772.2, 1772.5, 1745.2, 1690.2, 1624.9, 1589.0, 1618.5,
     R 1701.3, 1783.2, 1816.4, 1800.7, 1765.0, 1734.1, 1714.6, 1705.0 /
C        SOLAR SPECTRUM FROM  23860 TO  26720 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA10  /
     A 1701.6, 1696.6, 1682.0, 1661.4, 1657.2, 1693.0, 1763.2, 1826.5,
     B 1841.6, 1806.1, 1755.6, 1725.8, 1724.2, 1736.8, 1749.0, 1756.1,
     C 1759.5, 1762.1, 1770.2, 1791.7, 1826.8, 1848.9, 1819.6, 1720.7,
     D 1595.5, 1513.9, 1522.5, 1602.0, 1706.2, 1793.4, 1837.9, 1820.3,
     E 1738.3, 1631.1, 1553.1, 1539.2, 1574.3, 1623.9, 1660.6, 1676.8,
     F 1673.1, 1652.9, 1626.4, 1606.7, 1604.2, 1620.9, 1654.5, 1701.2,
     G 1752.2, 1796.2, 1822.8, 1827.4, 1808.5, 1767.0, 1713.9, 1667.3,
     H 1643.7, 1643.5, 1652.5, 1655.3, 1638.7, 1592.2, 1506.4, 1377.3,
     I 1209.5, 1010.5, 807.59, 666.84, 664.53, 835.23, 1099.6, 1330.7,
     J 1423.2, 1363.7, 1194.1, 961.77, 725.04, 551.29, 504.01, 596.30,
     K 775.15, 975.62, 1150.2, 1287.2, 1386.1, 1447.5, 1473.7, 1468.5,
     L 1435.2, 1376.9, 1296.0, 1195.5, 1085.3, 985.40, 917.25, 894.59,
     M 910.86, 951.53, 1001.7, 1046.4, 1070.7, 1061.2, 1021.2, 977.16,
     N 959.15, 982.06, 1020.5, 1032.6, 983.44, 879.83, 762.66, 675.28,
     O 643.33, 662.65, 721.49, 808.35, 913.24, 1027.0, 1139.9, 1236.2,
     P 1293.2, 1287.1, 1210.4, 1102.1, 1021.6, 1022.8, 1109.3, 1232.6,
     Q 1337.0, 1383.1, 1372.8, 1324.7, 1257.7, 1188.8, 1133.5, 1106.5,
     R 1113.7, 1136.8, 1147.9, 1121.4, 1054.1, 968.10, 889.19, 837.87 /
C        SOLAR SPECTRUM FROM  26740 TO  28780 CM-1,  IN STEPS OF 20 CM-
      DATA SUNA11  /
     A 817.64, 823.72, 851.04, 896.53, 959.85, 1041.2, 1137.6, 1231.2,
     B 1294.4, 1299.9, 1241.2, 1155.0, 1092.0, 1097.1, 1170.2, 1263.5,
     C 1322.4, 1307.4, 1233.6, 1146.1, 1090.8, 1092.5, 1134.6, 1188.9,
     D 1228.9, 1245.5, 1248.5, 1250.3, 1260.5, 1274.6, 1279.5, 1261.8,
     E 1214.3, 1145.4, 1069.6, 1001.4, 952.52, 930.48, 941.68, 990.34,
     F 1064.4, 1135.2, 1171.5, 1149.1, 1076.3, 984.35, 906.25, 868.17,
     G 873.75, 915.33, 984.41, 1067.2, 1137.1, 1163.1, 1115.5, 990.55,
     H 830.93, 692.29, 627.44, 654.10, 739.24, 838.88, 911.69, 941.90,
     I 944.42, 939.58, 946.10, 970.23, 1005.2, 1042.4, 1073.8, 1097.0,
     J 1114.3, 1128.8, 1142.9, 1153.4, 1152.4, 1131.5, 1084.2, 1016.7,
     K 945.95, 890.37, 866.15, 876.54, 913.13, 966.10, 1025.4, 1080.2,
     L 1119.0, 1102.7, 1243.5, 1209.9, 1079.2, 852.20, 956.80, 842.31,
     M 897.44, 1081.8, 914.23, 993.09, 1049.8, 844.95, 839.16/
C        SOLAR SPECTRUM FROM  28400 TO  29830 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB01  /
     A 876.54, 892.17, 913.13, 938.18, 966.10, 995.62, 1025.4, 1054.1,
     B 1080.2, 1102.1, 1119.0, 1132.2, 1102.7, 1159.3, 1243.5, 1238.3,
     C 1209.9, 1196.2, 1079.2, 895.60, 852.20, 935.59, 956.80, 897.09,
     D 842.31, 821.15, 897.44, 1042.7, 1081.8, 988.79, 914.23, 929.38,
     E 993.09, 1041.9, 1049.8, 984.33, 844.95, 770.76, 839.16, 939.65,
     F 1026.1, 1121.1, 1162.6, 1142.6, 1077.9, 1027.3, 1078.2, 1094.3,
     G 969.83, 853.72, 849.91, 909.12, 995.68, 1095.0, 1146.9, 1086.3,
     H 1010.4, 1065.4, 1128.9, 1080.6, 987.93, 898.18, 835.20, 771.63,
     I 687.12, 614.52, 606.14, 737.09, 908.13, 997.64, 1080.6, 1126.3,
     J 1056.7, 1028.4, 1141.7, 1252.6, 1225.3, 1103.2, 1038.6, 1043.4,
     K 1002.9, 965.51, 1035.0, 1150.7, 1200.9, 1152.0, 1068.5, 995.84,
     L 889.52, 818.48, 907.01, 1042.2, 1055.6, 1000.6, 972.00, 985.72,
     M 1027.2, 1054.8, 1078.0, 1126.6, 1205.3, 1245.7, 1201.0, 1144.7,
     N 1097.5, 1030.1, 926.85, 836.71, 864.11, 993.50, 1075.3, 1032.6,
     O 1008.9, 1066.1, 1067.4, 1004.8, 971.54, 923.18, 815.71, 799.70,
     P 946.19, 1100.1, 1126.4, 1032.2, 895.14, 784.30, 734.77, 726.53,
     Q 726.88, 765.54, 863.90, 992.24, 1070.9, 1028.1, 858.78, 647.15,
     R 563.18, 679.98, 906.40, 1094.3, 1155.3, 1124.3, 1098.4, 1109.5 /
C        SOLAR SPECTRUM FROM  29840 TO  31270 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB02  /
     A 1076.2, 944.17, 849.20, 928.54, 1062.0, 1118.9, 1119.2, 1074.6,
     B 1005.8, 980.02, 999.11, 1002.4, 939.78, 838.12, 816.13, 908.73,
     C 1014.9, 1058.3, 1043.7, 987.54, 946.35, 981.40, 1055.8, 1094.3,
     D 1028.3, 916.41, 908.99, 991.83, 1049.6, 1076.2, 1093.5, 1076.3,
     E 1014.5, 949.61, 947.26, 1001.2, 1051.5, 1072.8, 1068.0, 1012.5,
     F 907.81, 866.30, 950.89, 1037.5, 1079.5, 1183.9, 1291.3, 1268.6,
     G 1199.3, 1188.6, 1188.0, 1186.6, 1198.2, 1171.3, 1132.6, 1131.6,
     H 1096.0, 971.10, 847.07, 836.62, 922.78, 990.99, 987.51, 969.24,
     I 981.46, 981.36, 971.95, 985.34, 1003.0, 1037.2, 1071.2, 1065.7,
     J 1026.7, 984.84, 1002.7, 1070.3, 1117.5, 1116.0, 1048.9, 965.34,
     K 972.27, 1045.7, 1096.6, 1127.5, 1133.5, 1099.6, 1079.3, 1082.9,
     L 1026.8, 927.50, 879.08, 858.83, 831.01, 807.82, 789.56, 813.75,
     M 893.46, 937.62, 901.56, 864.46, 873.35, 891.03, 862.46, 810.30,
     N 787.36, 752.93, 715.34, 708.07, 728.93, 786.79, 807.73, 736.28,
     O 645.08, 616.90, 649.17, 691.77, 749.18, 820.21, 820.68, 791.26,
     P 854.27, 940.56, 956.38, 909.42, 824.18, 767.17, 722.06, 653.42,
     Q 624.67, 633.73, 655.14, 707.93, 784.94, 880.79, 961.15, 985.60,
     R 986.18, 966.53, 921.47, 888.89, 855.85, 851.66, 886.78, 850.97 /
C        SOLAR SPECTRUM FROM  31280 TO  32710 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB03  /
     A 766.97, 738.95, 724.53, 657.61, 587.77, 616.86, 760.61, 903.23,
     B 917.27, 838.49, 784.80, 759.41, 719.61, 671.48, 624.63, 588.57,
     C 574.70, 596.68, 698.02, 866.39, 974.82, 960.37, 930.10, 962.65,
     D 1007.1, 1001.9, 926.29, 816.64, 763.25, 772.93, 762.66, 729.39,
     E 725.01, 727.16, 672.73, 581.42, 520.97, 488.80, 478.60, 542.08,
     F 663.71, 749.48, 785.87, 811.05, 818.19, 813.80, 824.54, 836.62,
     G 799.66, 728.00, 660.36, 559.28, 473.28, 550.16, 752.04, 885.84,
     H 906.80, 912.21, 929.32, 899.72, 830.20, 774.56, 736.42, 724.09,
     I 740.12, 754.11, 764.96, 780.76, 788.94, 784.87, 758.80, 725.91,
     J 751.84, 804.24, 777.73, 703.36, 665.27, 663.99, 679.36, 706.09,
     K 757.57, 836.09, 880.02, 881.18, 907.91, 929.26, 894.32, 874.01,
     L 918.56, 953.50, 922.32, 866.61, 836.54, 825.28, 752.54, 586.02,
     M 427.46, 374.05, 437.23, 534.32, 556.74, 563.11, 629.31, 631.26,
     N 518.76, 438.31, 460.31, 530.45, 608.50, 657.99, 662.08, 686.17,
     O 775.18, 843.11, 797.46, 685.33, 611.33, 628.74, 711.36, 754.94,
     P 728.80, 722.79, 726.38, 679.68, 665.83, 710.48, 723.10, 724.09,
     Q 760.18, 784.01, 742.78, 634.33, 546.55, 563.54, 611.03, 623.16,
     R 665.36, 743.55, 764.46, 671.14, 513.18, 401.86, 405.77, 515.72 /
C        SOLAR SPECTRUM FROM  32720 TO  34150 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB04  /
     A 639.90, 677.85, 679.55, 759.33, 848.11, 819.89, 751.75, 710.50,
     B 615.33, 525.09, 583.35, 715.23, 767.53, 739.10, 664.05, 580.57,
     C 572.85, 634.13, 648.77, 561.27, 497.72, 591.71, 737.83, 794.19,
     D 802.51, 799.33, 735.79, 658.41, 659.47, 718.18, 761.67, 697.24,
     E 545.14, 474.47, 526.96, 597.65, 584.74, 447.28, 291.35, 261.28,
     F 330.26, 401.96, 466.32, 531.26, 572.34, 584.86, 585.17, 569.46,
     G 558.27, 559.41, 512.02, 426.37, 378.14, 398.26, 473.49, 542.18,
     H 531.76, 437.48, 341.85, 305.82, 299.88, 328.12, 440.04, 586.46,
     I 660.32, 625.22, 510.26, 418.85, 447.36, 534.89, 605.86, 667.07,
     J 687.31, 636.79, 549.63, 472.88, 419.53, 370.06, 327.98, 320.49,
```

```
     K 354.00, 399.17, 450.98, 528.34, 608.25, 696.07, 774.28, 760.75,
     L 690.58, 648.20, 580.61, 477.96, 453.91, 488.74, 464.02, 421.59,
     M 444.32, 446.59, 375.95, 342.13, 397.49, 510.97, 646.38, 725.14,
     N 703.06, 639.06, 619.10, 654.66, 665.99, 611.40, 580.22, 607.29,
     O 591.05, 542.30, 583.82, 673.02, 673.21, 582.44, 465.73, 377.25,
     P 377.04, 487.27, 607.93, 617.52, 583.46, 601.68, 615.94, 575.47,
     Q 541.63, 542.06, 522.28, 472.49, 423.29, 438.09, 556.72, 664.34,
     R 669.88, 657.45, 684.71, 705.70, 683.11, 600.81, 509.90, 497.64 /
C         SOLAR SPECTRUM FROM  34160 TO  35590 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB05  /
     A 511.07, 496.07, 500.32, 518.70, 529.91, 563.00, 609.20, 626.49,
     B 622.11, 615.72, 600.44, 591.26, 598.12, 593.07, 590.94, 631.58,
     C 696.48, 718.48, 676.11, 631.56, 619.64, 620.53, 624.10, 636.56,
     D 658.02, 688.78, 724.81, 742.60, 722.31, 675.86, 665.96, 704.73,
     E 703.70, 645.00, 598.26, 587.77, 590.94, 575.93, 528.03, 477.92,
     F 457.52, 456.80, 454.91, 448.65, 445.47, 445.38, 444.43, 446.04,
     G 455.91, 468.02, 454.34, 393.32, 301.22, 211.44, 167.11, 193.99,
     H 254.01, 305.35, 353.03, 385.08, 387.03, 391.60, 406.20, 415.34,
     I 435.34, 469.77, 492.15, 472.73, 409.86, 353.25, 340.68, 355.27,
     J 379.77, 401.81, 409.67, 406.89, 393.16, 378.89, 375.20, 373.52,
     K 360.19, 322.79, 273.55, 237.76, 212.33, 184.80, 156.20, 127.75,
     L 96.269, 68.806, 62.047, 77.143, 100.47, 127.56, 159.88, 194.05,
     M 225.20, 254.64, 285.75, 300.14, 294.40, 308.92, 340.83, 346.26,
     N 336.29, 347.54, 373.81, 388.78, 372.68, 325.29, 294.40, 317.56,
     O 360.30, 378.08, 374.22, 374.03, 383.34, 387.88, 377.55, 356.96,
     P 340.67, 328.71, 314.00, 316.91, 344.51, 355.54, 335.66, 318.68,
     Q 318.65, 322.43, 318.61, 304.92, 284.84, 268.13, 265.80, 273.55,
     R 274.18, 252.38, 215.04, 188.60, 181.31, 181.31, 180.78, 175.24 /
C         SOLAR SPECTRUM FROM  35600 TO  37030 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB06  /
     A 162.06, 145.08, 128.76, 113.76, 98.078, 83.072, 76.222, 78.359,
     B 78.434, 74.235, 75.843, 80.321, 77.859, 70.298, 64.651, 67.049,
     C 77.810, 83.167, 75.286, 71.202, 80.549, 92.008, 100.17, 108.63,
     D 119.44, 130.78, 142.31, 158.94, 177.12, 186.40, 186.60, 181.47,
     E 175.30, 175.54, 179.00, 177.04, 172.60, 172.67, 178.98, 193.77,
     F 215.13, 233.62, 252.05, 277.68, 298.91, 298.40, 280.81, 274.21,
     G 286.52, 285.46, 259.71, 241.39, 246.98, 259.87, 274.27, 298.47,
     H 316.85, 303.19, 263.69, 229.31, 227.90, 256.12, 281.58, 300.19,
     I 310.56, 279.54, 211.93, 152.18, 129.94, 147.47, 181.62, 215.37,
     J 239.50, 233.12, 191.55, 139.41, 110.51, 118.93, 134.79, 129.05,
     K 124.39, 143.53, 158.29, 141.84, 116.32, 111.59, 128.93, 149.17,
     L 153.44, 145.63, 148.52, 159.25, 155.84, 154.17, 177.28, 203.40,
     M 207.35, 205.27, 222.85, 253.18, 271.28, 279.27, 302.17, 321.47,
     N 288.83, 230.14, 206.40, 213.22, 216.49, 207.46, 196.20, 195.21,
     O 202.03, 194.33, 164.86, 136.65, 123.87, 128.14, 161.89, 216.99,
     P 253.68, 249.26, 222.89, 213.11, 243.64, 293.10, 309.42, 286.40,
     Q 269.61, 272.23, 271.67, 265.84, 265.61, 264.77, 266.03, 289.51,
     R 325.67, 337.34, 321.17, 300.30, 282.60, 287.14, 322.06, 335.79 /
C         SOLAR SPECTRUM FROM  37040 TO  38470 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB07  /
     A 297.22, 254.10, 243.47, 239.49, 219.32, 211.94, 239.28, 271.43,
     B 279.37, 272.26, 264.77, 250.52, 229.93, 222.15, 235.30, 256.79,
     C 275.28, 286.92, 284.85, 269.52, 255.05, 253.46, 263.22, 274.78,
     D 279.19, 270.17, 249.41, 229.04, 221.64, 231.38, 252.70, 280.64,
     E 310.06, 328.33, 325.01, 290.26, 238.97, 223.38, 257.24, 282.60,
     F 264.32, 243.34, 253.18, 272.89, 271.32, 256.12, 260.24, 271.35,
     G 257.11, 236.61, 238.72, 248.92, 255.90, 272.04, 291.78, 297.40,
     H 288.09, 283.28, 292.92, 301.74, 309.07, 322.05, 320.42, 295.43,
     I 269.65, 254.41, 240.88, 228.18, 221.23, 213.72, 201.23, 197.17,
     J 212.29, 233.39, 247.65, 261.74, 286.17, 322.49, 349.47, 338.28,
     K 297.06, 261.55, 252.28, 264.65, 286.92, 298.94, 280.45, 244.37,
     L 213.47, 193.03, 182.07, 168.54, 143.12, 114.10, 89.615, 73.589,
     M 73.990, 87.912, 96.265, 94.813, 96.604, 102.30, 102.15, 103.07,
     N 117.81, 137.41, 146.09, 144.28, 137.89, 128.11, 122.82, 128.19,
     O 130.66, 117.31, 98.912, 93.397, 105.63, 122.73, 126.39, 113.05,
     P 92.317, 76.340, 69.032, 66.324, 71.280, 87.431, 105.94, 114.02,
     Q 107.91, 91.872, 75.208, 69.123, 75.930, 90.928, 109.71, 125.70,
     R 135.79, 141.14, 138.14, 121.33, 91.806, 63.497, 52.106, 59.555 /
C         SOLAR SPECTRUM FROM  38480 TO  39910 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB08  /
     A 81.015, 106.67, 118.97, 116.36, 110.82, 100.88, 89.056, 90.431,
     B 104.41, 114.95, 124.85, 148.87, 171.72, 167.22, 142.25, 118.42,
     C 98.653, 78.908, 68.133, 77.286, 100.93, 120.08, 125.49, 131.79,
     D 155.69, 180.75, 181.81, 166.77, 150.06, 133.24, 116.14, 97.728,
     E 81.629, 76.695, 87.607, 110.23, 134.88, 149.13, 147.64, 139.88,
     F 135.19, 135.07, 138.00, 136.73, 128.84, 122.22, 120.48, 121.98,
     G 123.08, 116.30, 101.43, 86.303, 74.719, 68.800, 71.327, 80.626,
     H 90.485, 96.739, 100.69, 100.81, 93.677, 84.740, 81.532, 82.893,
     I 84.564, 87.584, 91.780, 91.272, 87.014, 87.386, 90.149, 84.917,
     J 71.266, 57.873, 51.863, 53.876, 57.909, 58.508, 57.020, 57.432,
     K 60.671, 64.667, 67.362, 67.511, 64.233, 59.035, 55.697, 56.636,
     L 59.400, 59.070, 56.522, 55.834, 55.860, 54.039, 51.976, 52.344,
     M 54.667, 56.450, 56.751, 56.769, 58.002, 60.029, 59.602, 53.134,
     N 42.926, 35.588, 33.447, 35.171, 39.379, 44.371, 47.745, 46.933,
     O 42.441, 37.879, 35.595, 36.458, 41.048, 47.300, 51.098, 50.024,
     P 45.331, 41.282, 40.082, 40.000, 39.104, 37.329, 36.632, 37.792,
     Q 39.189, 41.058, 45.214, 50.737, 54.281, 55.015, 56.138, 60.931,
     R 67.383, 69.534, 65.159, 56.372, 47.326, 44.322, 49.944, 59.696 /
C         SOLAR SPECTRUM FROM  39920 TO  41350 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB09  /
     A 67.929, 71.334, 69.905, 65.620, 59.303, 54.016, 55.880, 65.155,
     B 74.065, 76.217, 73.506, 71.406, 70.849, 69.749, 69.268, 71.380,
     C 72.721, 68.929, 61.665, 54.896, 47.420, 38.325, 32.219, 31.243,
     D 33.310, 35.358, 35.623, 36.840, 41.551, 47.499, 51.176, 50.344,
     E 45.362, 38.341, 33.130, 33.801, 40.140, 49.121, 55.385, 55.174,
     F 50.450, 46.511, 47.495, 51.883, 56.354, 59.603, 61.584, 63.215,
     G 64.603, 64.101, 59.027, 50.956, 47.633, 52.543, 58.883, 59.829,
     H 57.617, 56.727, 57.371, 57.898, 57.177, 55.129, 52.952, 52.018,
     I 52.186, 52.044, 50.269, 46.592, 42.515, 40.755, 41.887, 44.119,
     J 46.536, 48.858, 50.490, 51.919, 54.085, 54.707, 51.927, 49.449,
     K 49.865, 50.933, 50.496, 48.616, 46.717, 46.070, 46.263, 46.733,
     L 48.009, 50.187, 52.420, 53.536, 52.507, 51.380, 53.214, 56.985,
     M 60.614, 63.139, 63.999, 63.869, 65.100, 69.385, 74.743, 78.184,
     N 78.103, 74.113, 67.371, 60.849, 58.924, 62.682, 68.032, 69.117,
     O 64.604, 59.110, 55.998, 56.838, 61.778, 65.874, 65.079, 63.038,
     P 64.809, 69.911, 74.841, 76.439, 73.587, 68.853, 67.497, 72.675,
     Q 80.602, 83.422, 78.957, 72.228, 66.737, 62.842, 61.535, 63.574,
     R 69.248, 76.577, 79.922, 77.755, 73.938, 70.518, 68.003, 66.339 /
C         SOLAR SPECTRUM FROM  41360 TO  42790 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB10  /
     A 63.979, 61.098, 59.421, 58.103, 55.741, 52.549, 48.079, 42.578,
     B 38.373, 37.297, 37.455, 34.861, 30.483, 29.634, 34.734, 42.460,
     C 47.066, 45.848, 40.157, 34.290, 31.584, 30.650, 29.054, 27.788,
     D 30.427, 37.570, 44.196, 46.880, 47.848, 49.166, 49.180, 45.002,
     E 38.135, 35.055, 38.095, 41.750, 40.899, 35.722, 28.884, 24.835,
     F 28.670, 39.646, 50.310, 55.725, 57.401, 58.110, 59.406, 59.360,
     G 53.420, 43.004, 34.787, 33.697, 39.682, 47.554, 52.605, 53.632,
     H 51.001, 45.266, 37.844, 31.030, 25.936, 22.799, 21.882, 23.484,
     I 27.857, 33.447, 37.319, 39.195, 42.826, 50.398, 58.752, 63.301,
     J 61.094, 53.532, 46.046, 41.118, 37.646, 36.304, 40.426, 50.893,
     K 61.553, 65.395, 62.680, 58.087, 54.622, 51.330, 46.874, 42.870,
     L 40.547, 39.760, 40.217, 40.359, 39.559, 40.667, 46.260, 53.413,
     M 56.041, 52.566, 46.674, 41.073, 35.511, 31.231, 31.082, 35.955,
     N 45.199, 55.464, 61.802, 63.505, 61.850, 56.412, 49.388, 46.369,
     O 50.058, 56.694, 60.884, 61.030, 58.107, 54.303, 51.940, 50.508,
     P 46.749, 39.155, 31.535, 28.959, 30.973, 32.670, 31.567, 29.340,
     Q 27.275, 25.184, 24.264, 27.068, 34.296, 42.475, 47.230, 47.425,
     R 44.435, 40.538, 36.868, 33.020, 29.405, 28.753, 34.079, 44.246 /
C         SOLAR SPECTRUM FROM  42800 TO  44230 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB11  /
     A 53.780, 57.974, 56.376, 51.200, 45.308, 40.273, 35.900, 33.344,
     B 34.011, 36.858, 41.283, 47.374, 53.088, 56.201, 55.633, 50.843,
     C 43.997, 38.767, 36.248, 36.380, 40.762, 50.700, 63.371, 73.432,
     D 76.418, 70.373, 58.741, 47.034, 38.598, 34.664, 35.794, 42.084,
     E 49.973, 54.338, 53.956, 52.287, 52.778, 55.571, 59.034, 60.268,
     F 56.247, 47.362, 38.056, 32.889, 31.739, 31.734, 32.476, 35.060,
```

```
     G 39.091, 43.398, 48.131, 51.574, 58.749, 63.599, 68.971, 73.421,
     H 73.861, 69.003, 60.557, 51.865, 44.879, 42.060, 44.802, 47.950,
     I 46.882, 42.973, 39.293, 37.711, 37.137, 35.222, 32.243, 30.488,
     J 32.605, 40.429, 51.099, 57.710, 57.150, 52.992, 50.275, 49.986,
     K 49.778, 48.371, 46.421, 44.604, 42.730, 41.244, 41.565, 43.805,
     L 47.013, 48.992, 46.428, 40.595, 37.840, 42.353, 52.248, 60.529,
     M 61.566, 56.800, 52.041, 52.260, 57.077, 61.019, 60.712, 57.048,
     N 51.481, 46.352, 44.366, 44.947, 45.478, 44.944, 43.825, 42.105,
     O 39.466, 36.826, 35.907, 36.357, 35.661, 33.947, 33.690, 34.429,
     P 34.000, 32.645, 31.410, 30.281, 29.409, 29.127, 29.326, 29.869,
     Q 30.601, 31.311, 32.099, 32.779, 32.757, 32.098, 31.975, 33.484,
     R 36.048, 39.169, 43.365, 47.244, 48.214, 45.786, 41.586, 38.775 /
C         SOLAR SPECTRUM FROM  44240 TO  45670 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB12  /
     A 40.753, 46.752, 51.684, 52.597, 51.449,. 50.684, 49.450, 46.747,
     B 45.369, 47.685, 50.240, 48.961, 46.693, 48.600, 53.694, 56.465,
     C 54.341, 50.722, 49.877, 51.246, 52.088, 52.765, 56.254, 63.326,
     D 69.744, 71.066, 68.349, 65.123, 62.551, 59.195, 53.705, 48.161,
     E 46.236, 47.710, 49.660, 50.799, 51.836, 54.537, 59.647, 64.707,
     F 65.844, 61.634, 55.570, 54.083, 58.781, 64.888, 69.777, 74.008,
     G 76.492, 76.226, 74.746, 74.941, 77.801, 79.619, 76.190, 67.190,
     H 55.231, 45.813, 43.141, 45.647, 49.466, 52.231, 52.221, 48.886,
     I 44.716, 42.613, 43.385, 45.968, 48.121, 48.998, 49.885, 50.707,
     J 49.893, 48.319, 48.198, 50.280, 53.830, 55.914, 54.822, 52.939,
     K 51.944, 49.438, 42.956, 34.614, 28.100, 24.503, 24.203, 27.839,
     L 34.604, 41.615, 45.324, 45.444, 45.527, 47.179, 45.756, 36.862,
     M 26.037, 20.569, 20.329, 24.263, 30.863, 35.939, 36.711, 35.693,
     N 37.256, 40.862, 44.416, 48.800, 54.182, 57.655, 58.427, 59.965,
     O 63.940, 66.820, 65.465, 59.482, 49.396, 39.422, 34.182, 35.388,
     P 42.875, 52.034, 57.595, 59.093, 57.272, 52.172, 45.493, 39.419,
     Q 35.581, 35.902, 40.354, 46.732, 53.309, 58.781, 61.785, 59.255,
     R 50.030, 41.567, 40.523, 43.584, 44.875, 42.754, 40.077, 39.941 /
C         SOLAR SPECTRUM FROM  45680 TO  47110 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB13  /
     A 40.977, 39.567, 34.955, 30.424, 31.039, 38.687, 47.480, 49.830,
     B 46.790, 44.829, 46.546, 50.415, 54.602, 57.656, 58.463, 57.276,
     C 55.621, 54.514, 53.338, 50.026, 42.817, 33.636, 27.134, 25.516,
     D 27.897, 31.392, 32.125, 29.463, 26.581, 25.956, 27.737, 31.175,
     E 34.959, 37.671, 38.641, 37.958, 36.733, 35.681, 33.877, 30.849,
     F 28.059, 27.615, 29.319, 29.375, 25.390, 20.659, 19.484, 22.297,
     G 27.282, 32.467, 35.906, 37.137, 37.895, 39.130, 39.777, 39.872,
     H 40.778, 42.317, 42.934, 40.430, 34.227, 27.701, 23.880, 22.174,
     I 21.639, 22.589, 25.184, 29.017, 32.981, 36.110, 38.580, 41.239,
     J 44.426, 46.939, 47.010, 44.165, 39.659, 35.556, 32.838, 31.546,
     K 32.676, 36.963, 42.333, 44.931, 43.704, 40.943, 37.973, 35.199,
     L 33.574, 33.339, 34.185, 36.347, 39.963, 43.964, 47.162, 48.987,
     M 48.976, 47.948, 48.004, 49.892, 51.065, 47.834, 40.489, 32.665,
     N 26.795, 24.461, 26.655, 31.928, 37.634, 41.345, 40.956, 36.827,
     O 32.110, 28.612, 26.482, 26.602, 28.831, 30.877, 30.976, 30.063,
     P 29.887, 30.305, 29.974, 28.265, 26.517, 27.066, 30.403, 34.539,
     Q 37.104, 37.598, 37.252, 37.060, 36.498, 34.167, 29.814, 24.192,
     R 18.515, 15.086, 15.040, 17.158, 20.807, 25.682, 30.352, 34.203 /
C         SOLAR SPECTRUM FROM  47120 TO  48550 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB14  /
     A 37.902, 42.531, 47.832, 50.509, 48.019, 42.616, 38.321, 37.370,
     B 40.172, 44.395, 46.132, 43.911, 38.396, 31.379, 26.275, 25.075,
     C 26.652, 28.963, 31.168, 34.168, 38.050, 40.231, 38.347, 32.741,
     D 26.199, 21.863, 20.249, 20.185, 21.726, 25.562, 30.318, 33.431,
     E 34.453, 34.959, 36.374, 37.870, 36.655, 31.966, 25.920, 21.264,
     F 20.663, 24.658, 30.263, 34.021, 34.336, 31.356, 26.926, 23.109,
     G 20.867, 20.684, 22.416, 24.878, 26.779, 27.334, 26.537, 25.210,
     H 24.013, 22.944, 21.800, 20.449, 19.290, 19.528, 21.742, 24.125,
     I 23.994, 21.559, 19.555, 18.915, 18.342, 17.335, 16.549, 16.479,
     J 17.211, 18.445, 19.294, 18.980, 17.912, 17.156, 17.103, 17.256,
     K 16.925, 15.842, 14.485, 13.683, 13.647, 13.914, 14.009, 13.770,
     L 13.456, 13.399, 13.547, 13.760, 14.060, 14.427, 14.644, 14.438,
     M 13.986, 13.749, 13.927, 14.390, 14.759, 14.822, 14.679, 14.448,
     N 14.186, 13.937, 13.754, 13.657, 13.540, 13.308, 13.053, 12.841,
     O 12.704, 12.742, 12.811, 12.662, 12.355, 12.100, 12.003, 12.014,
     P 12.067, 12.223, 12.444, 12.472, 12.164, 11.732, 11.515, 11.619,
     Q 11.873, 12.028, 11.947, 11.722, 11.399, 10.930, 10.473, 10.205,
     R 10.224, 10.694, 11.468, 12.007, 12.083, 11.905, 11.498, 10.891 /
C         SOLAR SPECTRUM FROM  48560 TO  49990 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB15  /
     A 10.575, 10.846, 11.353, 11.612, 11.411, 10.876, 10.383, 10.305,
     B 10.695, 11.245, 11.636, 11.828, 11.918, 11.865, 11.674, 11.510,
     C 11.407, 11.303, 11.216, 11.143, 11.039, 10.983, 11.004, 10.900,
     D 10.653, 10.562, 10.781, 11.186, 11.605, 11.806, 11.582, 11.056,
     E 10.567, 10.335, 10.408, 10.729, 11.165, 11.540, 11.646, 11.372,
     F 10.933, 10.524, 9.9973, 9.3783, 8.9883, 9.0163, 9.4125, 9.9179,
     G 10.278, 10.472, 10.553, 10.575, 10.519, 10.216, 9.6821, 9.1499,
     H 8.7057, 8.3894, 8.3442, 8.6241, 9.1371, 9.7184, 10.191, 10.443,
     I 10.458, 10.289, 9.9772, 9.5829, 9.3097, 9.3195, 9.4694, 9.5182,
     J 9.4326, 9.2478, 8.8197, 7.9809, 6.9996, 6.4856, 6.7462, 7.5406,
     K 8.2813, 8.7258, 9.0682, 9.1665, 8.8637, 8.4638, 8.2393, 8.1656,
     L 8.1880, 8.3578, 8.6488, 8.8980, 9.0117, 9.0659, 9.1955, 9.4207,
     M 9.5526, 9.4237, 9.1290, 8.8441, 8.6138, 8.4237, 8.2979, 8.2598,
     N 8.2859, 8.3475, 8.4533, 8.6285, 8.8310, 8.8866, 8.6750, 8.3312,
     O 8.0091, 7.7296, 7.6239, 7.8692, 8.2725, 8.4086, 8.2515, 8.0914,
     P 8.0003, 7.9367, 7.9266, 7.9580, 8.0492, 8.2376, 8.4263, 8.4811,
     Q 8.3309, 8.0263, 7.7632, 7.6987, 7.8124, 7.9390, 8.0183, 8.0816,
     R 8.0428, 7.8923, 7.6963, 7.4969, 7.4013, 7.4289, 7.4489, 7.4059 /
C         SOLAR SPECTRUM FROM  50000 TO  51430 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB16  /
     A 7.4198, 7.5261, 7.5252, 7.3239, 7.1263, 7.1423, 7.3340, 7.5049,
     B 7.5484, 7.5319, 7.5163, 7.4995, 7.5728, 7.8104, 8.0588, 8.0948,
     C 7.9140, 7.6978, 7.5116, 7.2138, 6.8063, 6.5430, 6.5232, 6.5869,
     D 6.5610, 6.3984, 6.1889, 6.0587, 6.0676, 6.1988, 6.3140, 6.2527,
     E 6.0929, 6.0277, 6.0941, 6.3031, 6.6594, 6.9398, 6.9566, 6.8310,
     F 6.7374, 6.6812, 6.6558, 6.8336, 7.2020, 7.4012, 7.2950, 7.0488,
     G 6.7966, 6.6293, 6.5868, 6.5980, 6.6007, 6.6501, 6.7627, 6.7853,
     H 6.6321, 6.4856, 6.5198, 6.6486, 6.7271, 6.7227, 6.6696, 6.6189,
     I 6.5979, 6.6188, 6.7110, 6.8343, 6.8750, 6.8250, 6.7885, 6.8266,
     J 6.8556, 6.8068, 6.8377, 7.0467, 7.2779, 7.4139, 7.4712, 7.4621,
     K 7.4071, 7.3592, 7.3372, 7.3220, 7.2938, 7.2531, 7.2052, 7.1335,
     L 7.0298, 6.8533, 6.5535, 6.2227, 6.0139, 5.9384, 5.9038, 5.8568,
     M 5.7909, 5.7326, 5.7745, 5.9608, 6.1865, 6.3681, 6.4997, 6.5437,
     N 6.4637, 6.2708, 6.0451, 5.9557, 6.0855, 6.2542, 6.2454, 6.0795,
     O 5.9102, 5.8447, 5.9218, 6.1063, 6.2895, 6.3271, 6.1097, 5.7421,
     P 5.4452, 5.2981, 5.3256, 5.4935, 5.6819, 5.8245, 5.8933, 5.9630,
     Q 6.1703, 6.4525, 6.6325, 6.6965, 6.7185, 6.6238, 6.3107, 5.9241,
     R 5.6987, 5.6651, 5.7428, 5.8790, 5.9715, 5.9618, 5.9674, 6.0754 /
C         SOLAR SPECTRUM FROM  51440 TO  52870 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB17  /
     A 6.2541, 6.4300, 6.4968, 6.4564, 6.4082, 6.3024, 6.0135, 5.6431,
     B 5.3963, 5.2989, 5.2635, 5.2227, 5.1279, 4.9315, 4.6348, 4.3168,
     C 4.0151, 3.6625, 3.2906, 3.1028, 3.1349, 3.1994, 3.2596, 3.4144,
     D 3.5949, 3.6534, 3.6296, 3.6281, 3.5876, 3.4292, 3.2659, 3.2284,
     E 3.2576, 3.3002, 3.4535, 3.7372, 4.0573, 4.3558, 4.5999, 4.7781,
     F 4.8855, 4.8999, 4.8392, 4.7624, 4.7059, 4.6981, 4.7666, 4.8453,
     G 4.8236, 4.7293, 4.6861, 4.7132, 4.7725, 4.8713, 4.9596, 4.9527,
     H 4.8957, 4.9252, 5.0736, 5.2229, 5.2505, 5.1537, 5.0156, 4.8880,
     I 4.7686, 4.6549, 4.5534, 4.4828, 4.4661, 4.5040, 4.5905, 4.7033,
     J 4.7858, 4.8334, 4.9283, 5.0377, 5.0065, 4.8471, 4.6828, 4.5586,
     K 4.4812, 4.4314, 4.3903, 4.3830, 4.4066, 4.3900, 4.2973, 4.1978,
     L 4.1462, 4.1084, 4.1495, 4.3897, 4.6859, 4.8206, 4.7938, 4.6781,
     M 4.5222, 4.3959, 4.3358, 4.2947, 4.2259, 4.1452, 4.1060, 4.1462,
     N 4.2149, 4.2549, 4.3061, 4.3742, 4.3738, 4.2718, 4.1389, 4.0405,
     O 3.9457, 3.8127, 3.7099, 3.7344, 3.8589, 3.9598, 3.9525, 3.8377,
     P 3.6708, 3.5357, 3.4929, 3.5375, 3.6381, 3.7890, 3.9671, 4.0995,
     Q 4.1421, 4.1302, 4.1235, 4.1623, 4.2506, 4.2948, 4.2231, 4.0993,
     R 3.9680, 3.9475, 4.1958, 4.5131, 4.6101, 4.5130, 4.3474, 4.1749 /
C         SOLAR SPECTRUM FROM  52880 TO  54310 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB18  /
     A 4.0467, 3.9956, 4.0078, 4.0374, 4.0255, 3.9379, 3.8192, 3.7529,
     B 3.7675, 3.8260, 3.8654, 3.8518, 3.8148, 3.8028, 3.8098, 3.7934,
```

```
     C 3.7660, 3.7944, 3.8689, 3.8978, 3.8856, 3.8923, 3.8570, 3.6940,
     D 3.4693, 3.3222, 3.2824, 3.2887, 3.3039, 3.3222, 3.3313, 3.3326,
     E 3.3482, 3.3807, 3.4188, 3.4602, 3.4972, 3.5151, 3.5155, 3.5165,
     F 3.5258, 3.5406, 3.5478, 3.5345, 3.5339, 3.5820, 3.6396, 3.6448,
     G 3.5872, 3.5112, 3.4804, 3.5257, 3.6238, 3.7290, 3.8023, 3.8024,
     H 3.7268, 3.6578, 3.6439, 3.6422, 3.6373, 3.6397, 3.6410, 3.6494,
     I 3.6608, 3.6251, 3.5212, 3.4020, 3.2845, 3.1230, 2.9483, 2.8515,
     J 2.8432, 2.8638, 2.8967, 2.9505, 3.0025, 3.0552, 3.1106, 3.1178,
     K 3.0596, 2.9854, 2.9316, 2.8903, 2.8590, 2.8500, 2.8450, 2.8121,
     L 2.7626, 2.7424, 2.7667, 2.8024, 2.8165, 2.8111, 2.8128, 2.8569,
     M 2.9659, 3.1062, 3.1990, 3.2128, 3.2088, 3.2391, 3.2661, 3.2364,
     N 3.1173, 2.9094, 2.6952, 2.5324, 2.3959, 2.2953, 2.2510, 2.2245,
     O 2.1811, 2.1301, 2.1482, 2.3257, 2.5856, 2.7226, 2.6495, 2.4508,
     P 2.2444, 2.0850, 1.9891, 1.9843, 2.0816, 2.2233, 2.3248, 2.3551,
     Q 2.3479, 2.3606, 2.4296, 2.5361, 2.6128, 2.6216, 2.6069, 2.6196,
     R 2.6464, 2.6427, 2.5823, 2.4682, 2.3320, 2.2405, 2.2637, 2.3973 /
C         SOLAR SPECTRUM FROM  54320 TO  55750 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB19  /
     A 2.5524, 2.6891, 2.8508, 3.0103, 3.0681, 3.0064, 2.9114, 2.8609,
     B 2.8517, 2.8374, 2.7894, 2.7288, 2.7138, 2.7729, 2.8707, 2.9536,
     C 2.9953, 2.9911, 2.9398, 2.8550, 2.7732, 2.7303, 2.7366, 2.7650,
     D 2.7705, 2.7374, 2.6830, 2.6218, 2.5663, 2.5341, 2.5351, 2.5681,
     E 2.6124, 2.6305, 2.6024, 2.5431, 2.4840, 2.4546, 2.4684, 2.5100,
     F 2.5445, 2.5532, 2.5564, 2.5889, 2.6616, 2.7553, 2.8466, 2.9290,
     G 2.9958, 3.0175, 2.9774, 2.8990, 2.8001, 2.6927, 2.6171, 2.5931,
     H 2.5809, 2.5276, 2.4284, 2.3365, 2.3162, 2.3855, 2.4872, 2.5455,
     I 2.5773, 2.6809, 2.9720, 3.5757, 4.4006, 5.0044, 5.0295, 4.5135,
     J 3.7071, 2.9059, 2.3600, 2.1418, 2.1119, 2.0871, 2.0301, 2.0043,
     K 2.0361, 2.0963, 2.1520, 2.1878, 2.1955, 2.1864, 2.1899, 2.2170,
     L 2.2574, 2.2895, 2.2783, 2.2148, 2.1641, 2.2343, 2.4726, 2.8119,
     M 3.1288, 3.2984, 3.2206, 2.8859, 2.4473, 2.1436, 2.0729, 2.1391,
     N 2.2171, 2.2580, 2.2654, 2.2481, 2.2103, 2.1657, 2.1356, 2.1321,
     O 2.1438, 2.1461, 2.1396, 2.1460, 2.1588, 2.1581, 2.1481, 2.1343,
     P 2.1101, 2.0754, 2.0400, 2.0121, 1.9930, 1.9799, 1.9699, 1.9613,
     Q 1.9537, 1.9454, 1.9312, 1.9058, 1.8726, 1.8470, 1.8465, 1.8693,
     R 1.8844, 1.8635, 1.8143, 1.7618, 1.7188, 1.6853, 1.6656, 1.6708 /
C         SOLAR SPECTRUM FROM  55760 TO  57190 CM-1,  IN STEPS OF 10 CM-
      DATA SUNB20  /
     A 1.7036, 1.7519, 1.8120, 1.9015, 2.0124, 2.0980, 2.1385, 2.1481,
     B 2.1347, 2.1086, 2.0953, 2.1062, 2.1095, 2.0685, 2.0001, 1.9461,
     C 1.9194, 1.9088, 1.9023, 1.8977, 1.9049, 1.9300, 1.9588, 1.9635,
     D 1.9357, 1.9019, 1.8887, 1.8939, 1.9018, 1.9038, 1.8975, 1.8747,
     E 1.8289, 1.7716, 1.7303, 1.7330, 1.7900, 1.8782, 1.9548, 1.9907,
     F 1.9807, 1.9430, 1.9173, 1.9218, 1.9203, 1.8717, 1.7832, 1.6965,
     G 1.6389, 1.6077, 1.5924, 1.5818, 1.5583, 1.5142, 1.4616, 1.4237,
     H 1.4252, 1.4834, 1.5970, 1.7410, 1.8771, 1.9784, 2.0451, 2.0872,
     I 2.0909, 2.0384, 1.9573, 1.9002, 1.8824, 1.8663, 1.8193, 1.7540,
     J 1.6874, 1.6222, 1.5726, 1.5450, 1.5290, 1.5312, 1.5699, 1.6411,
     K 1.7186, 1.7678, 1.7546, 1.6623, 1.5115, 1.3588, 1.2605, 1.2348,
     L 1.2611, 1.3091, 1.3588, 1.3884, 1.3800, 1.3482, 1.3224, 1.3159,
     M 1.3437, 1.4142, 1.4950, 1.5443, 1.5521, 1.5282, 1.4902, 1.4606,
     N 1.4465, 1.4398, 1.4399, 1.4544, 1.4760, 1.4781, 1.4506, 1.4229,
     O 1.4185, 1.4221, 1.4119, 1.3908, 1.3779, 1.3813, 1.3933, 1.4087,
     P 1.4268, 1.4417, 1.4408, 1.4188, 1.3861, 1.3548, 1.3261, 1.2980,
     Q 1.2769, 1.2731, 1.2856, 1.3002, 1.3056, 1.2987, 1.2817, 1.2590,
     R 1.2291, 1.1868, 1.1428, 1.1183, 1.1141, 1.1120, 1.1009, 1.0797 /
C      SOLAR SPECTRUM FROM  57200 TO  57490 CM-1, IN STEPS OF 10 CM-1
      DATA SUNB21  /
     A 1.0523, 1.0284, 1.0251, 1.0577, 1.1195, 1.1791, 1.2061, 1.2013,
     B 1.1936, 1.2000, 1.2040, 1.1824, 1.1489, 1.1400, 1.1539, 1.1629,
     C 1.1617, 1.1586, 1.1564, 1.1572, 1.1565, 1.1399, 1.1037, 1.0627,
     D 1.0341, 1.0223, 1.0199, 1.0188, 1.0174, 1.0163  /
      END
```

```
      subroutine spatial_var(nbands,nlcntx,necntx,diff,result)

c F77 ********************************************************
c
c ... routine for performing spatial variability tests and
c ... setting the appropriate flag.
c
c Input parameters:
c nbands        Total number MAS spectral bands
c nlcntx        Number scan lines in region (context).
c necntx        Number pixels in each scan of region (context).
c diff          Array of surrounding pixel reflectance or brightness
c               temperature differences from center pixel value
c Output Parameters:
c result        result  of spatial variability test (1=uniform)
c
c END*********************************************************

      INCLUDE 'thresholds.inc'

c ... get proper threshold value.
      parameter (masirll=11)'
c ...
c ... scalar arguments ..
      integer result,nbands,nlcntx,necntx
c ...
c ... array arguments ..
      real diff(nbands,nlcntx*necntx-1)
c ...
c ... local scalars ..
      integer i,ipt
c ...
c ... intrinsic functions ..
      intrinsic btest,ibset

      ipt = 0
      ndif = (nlcntx*necntx) - 1
c ... Compare surrounding bt differences to threshold.
      do 200 i = 1,ndif
        if (diff(masirll,i).le.dovarll(1)) then
          ipt = ipt + 1
        end if
  200 continue

c ... If all surrounding pixel differences were less than the
c ... threshold value, scene is declared to be uniform.
      if (ipt.eq.ndif) then
        result = 1
      else
        result = 0
      end if

      return
      end
```

```
      subroutine stats_out(nmpix,n1s,n2s,n3s,n4s)

c     Routine for calculating and writing output statistics.

      if(nmpix .gt. 0) then
         pcn1s = (float(n1s) / nmpix) * 100.0
         pcn2s = (float(n2s) / nmpix) * 100.0
         pcn3s = (float(n3s) / nmpix) * 100.0
         pcn4s = (float(n4s) / nmpix) * 100.0
      else
         pcn1s = 32767.0
         pcn2s = 32767.0
         pcn3s = 32767.0
         pcn4s = 32767.0
      end if

      write(*,'(1x,''stats: '')')

      write(*,'(1x,''# total pixels: '',i10)') nmpix
      write(*,'(1x,''# pixels > 99%: '',i10)') n1s
      write(*,'(1x,''# pixels > 95%: '',i10)') n2s
      write(*,'(1x,''# pixels > 66%: '',i10)') n3s
      write(*,'(1x,''# pixels <  1%: '',i10)') n4s
      write(*,'(1x)')

      write(*,'(1x,''% with confidence > 99% '',f10.5)') pcn1s
      write(*,'(1x,''% with confidence > 95% '',f10.5)') pcn2s
      write(*,'(1x,''% with confidence > 66% '',f10.5)') pcn3s
      write(*,'(1x,''% with confidence <  1% '',f10.5)') pcn4s

      return
      end
```

```
      integer function stop_proc(nelin,outf_unit,bitarray,iyrday,ibhms,
     *                     islhms1,outrec,npix,n1sm,n2sm,n3sm,n4sm,
     *                     if_hdfid,tp_unit,eco_unit,procflg)

c**********************************************************************

      include 'cldmask.inc'

c**********************************************************************
c
c       external variables

      integer*4 nelin           ! end line No.
      integer*4 outf_unit       ! unit No. of cloud flag file
      byte bitarray(2,npixel,*) ! cloud flag data
      integer*4 iyrday          ! date of first scan line (ddd)
      integer*4 ibhms           ! time of first scan line (hhmmss)
      integer*4 islhms1         ! start time of the cloud mask (hhmmss)
      integer*4 outrec(*)       ! output record No.
      integer*4 npix            ! # of total pixels
      integer*4 n1sm            ! # of pixels larger than 99%
      integer*4 n2sm            ! # of pixels larger than 95%
      integer*4 n3sm            ! # of pixels larger than 66%
      integer*4 n4sm            ! # of pixels larger than 1%
      integer*4 if_hdfid        ! ID No. of image data file
      integer*4 tp_unit         ! unit No. of land/sea flag file
      integer*4 eco_unit        ! unit No. of ecosystem file
      integer*4 procflg         ! procedure flag

c**********************************************************************
c
c       internal variables

      logical*4 end_file
      data end_file /.false./
      integer*4 line

c**********************************************************************

c           Write bit flags to output file for current scan line.

      if(procflg .eq. 0) then

c       write(*,'(1x,''#   file write '')')

c**********************************************************************

c     Write total number records output to the header record of the output
c     file.

         line = nelin + 1

c       write(*,'(1x,''#   end of file write '')')
c       write(*,'(1x,''#   outrec= '',i10)') outrec(line-1)
c       write(*,'(1x,''#   islhms1= '',i10)') islhms1

        end_file = .true.
        call write_bits(outf_unit,bitarray(1,1,linmax),iyrday,ibhms,
     *                  islhms1,outrec(line-1),end_file)

c**********************************************************************

c     Write out stats.

        call stats_out(npix,n1sm,n2sm,n3sm,n4sm)

      end if

c**********************************************************************

c     Close all files.

      call file_close(if_hdfid,tp_unit,eco_unit,
     *                outf_unit)

      return
      end
```

```
      SUBROUTINE TVIEW(KEY,XMU,BT11,CORR)
C
C
C     BI-DIMENSIONAL LINEAR OR QUADRATIC INTERPOLATION SCHEME ( LAGRANGE FORM )
C
C     INPUT PARAMETERS
C
C     KEY = 1 FOR LINEAR INTERPOLATION
C         = 2 FOR QUADRATIC INTERPOLATION
C     XMU = SECANT OF VIEWING ANGLE
C     BT11 = CHANNEL 4 BT
C
C     OUTPUT PARAMETERS
C
C     CORR = CORRECTION FACTOR FROM APOLLO
C
      REAL LU0,LU1,LU2,LT0,LT1,LT2
      DIMENSION UTAB(5),TTAB(6),TAB(5,6)
      DATA UTAB/2.00,1.75,1.5,1.25,1.00/
      DATA TTAB/260.,270.,280.,290.,300.,310./

      DATA (TAB(I, 1),I=1,5)/ ,
     * 1.10,0.90,0.65,0.60,0.55/
      DATA (TAB(I, 2),I=1,5)/
     * 1.13,1.03,0.81,0.63,0.58/
      DATA (TAB(I, 3),I=1,5)/
     * 2.30,2.14,1.88,1.61,1.30/
      DATA (TAB(I, 4),I=1,5)/
     * 4.73,4.27,3.95,3.72,3.06/
      DATA (TAB(I, 5),I=1,5)/
     * 8.43,7.42,7.00,6.92,5.77/
      DATA (TAB(I, 6),I=1,5)/
     * 13.39,11.60,11.03,10.74,9.41/

C
C     BOUNDS CHECK
C
      U=XMU
      T=BT11
      IF(U.GT.UTAB(1))U=UTAB(1)
      IF(U.LT.UTAB(5))U=UTAB(5)
      IF(T.LT.TTAB(1))T=TTAB(1)
      IF(T.GT.TTAB(6))T=TTAB(6)
C
C     SELECT THE XMU INDICIES
C
      DO 1 I=2,5
      II=I
      IF(U.GE.UTAB(I))GO TO 2
 1    CONTINUE
 2    IF(KEY.NE.1)GO TO 3
      I0=II-1
      I1=II
      GO TO 5
 3    IF(II.EQ.5)GO TO 4
      I0=II-1
      I1=II
      I2=II+1
      GO TO 5
 4    I0=II-2
      I1=II-1
      I2=II
C
C     SELECT THE BT11 INDICIES
C
 5    DO 6 J=2,6
      JJ=J
      IF(T.LE.TTAB(J))GO TO 7
 6    CONTINUE
 7    IF(KEY.NE.1)GO TO 8
      J0=JJ-1
      J1=JJ
      GO TO 10
 8    IF(JJ.EQ.6)GO TO 9
      J0=JJ-1
      J1=JJ
      J2=JJ+1
      GO TO 10
 9    J0=JJ-2
      J1=JJ-1
      J2=JJ
C
C     SET PARAMETER INDICIES
C
 10   continue
C
C     BRANCH ON SCHEME TYPE
C
      IF(KEY.NE.1)GO TO 20
C
C     LINEAR SCHEME
C
C     DESIGNATE INDEX VALUES
C
      U0=UTAB(I0)
      U1=UTAB(I1)
      T0=TTAB(J0)
      T1=TTAB(J1)
C
C     LAGRANGE POLYNOMIALS
C
      LU0=(U-U1)/(U0-U1)
      LU1=(U-U0)/(U1-U0)
      LT0=(T-T1)/(T0-T1)
      LT1=(T-T0)/(T1-T0)
C
C     LOOP OVER THE SCATTERING PARAMETER INDEX
C
C
C     INTERPOLATING POLYNOMIALS FOR THE FIRST DIMENSION
C
      P0=TAB(I0,J0)*LU0+TAB(I1,J0)*LU1
      P1=TAB(I0,J1)*LU0+TAB(I1,J1)*LU1
C
C     INTERPOLATING POLYNOMIAL FOR SECOND DIMENSION
C
      P=P0*LT0+P1*LT1
C
C     ASSIGN THE INDIVIDUAL PARAMETERS
C
      corr=p
 11   CONTINUE
      RETURN
C
C     QUADRATIC SCHEME
C
C     DESIGNATE INDEX VALUES
C
 20   U0=UTAB(I0)
      U1=UTAB(I1)
      U2=UTAB(I2)
      T0=TTAB(J0)
      T1=TTAB(J1)
      T2=TTAB(J2)
C
C     LAGRANGE POLYNOMIALS
```

```
C
      LU0=(U-U1)*(U-U2)/(U0-U1)/(U0-U2)
      LU1=(U-U0)*(U-U2)/(U1-U0)/(U1-U2)
      LU2=(U-U0)*(U-U1)/(U2-U0)/(U2-U1)
      LT0=(T-T1)*(T-T2)/(T0-T1)/(T0-T2)
      LT1=(T-T0)*(T-T2)/(T1-T0)/(T1-T2)
      LT2=(T-T0)*(T-T1)/(T2-T0)/(T2-T1)
C
C
C
C     INTERPOLATING POLYNOMIALS FOR THE FIRST DIMENSION
C
      P0=TAB(I0,J0)*LU0+TAB(I1,J0)*LU1+TAB(I2,J0)*LU2
      P1=TAB(I0,J1)*LU0+TAB(I1,J1)*LU1+TAB(I2,J1)*LU2
      P2=TAB(I0,J2)*LU0+TAB(I1,J2)*LU1+TAB(I2,J2)*LU2
C
C     INTERPOLATING POLYNOMIAL FOR SECOND DIMENSION
C
      P=P0*LT0+P1*LT1+P2*LT2
C
C     ASSIGN THE INDIVIDUAL PARAMETERS
C
      corr=P
 21   CONTINUE
      RETURN
      END
```

```
      subroutine water_day(pxldat,vza,pxrfa,snglnt,visusd,ice,
     *                     uniform,eco_type,pw,nlcntx,necntx,
     *                     inband,nbands,rgdata,avgtherm,
     *                     diff,testbits,confdnc)

c     Routine for setting appropriate flags and processing path
c     for daytime observations over water surfaces.

      integer*4 varslt
      real*4 pxldat(nbands),vza,confdnc,pw,avgtherm(nbands),
     *       rgdata(nlcntx,necntx,nbands),
     *       diff(nbands,nlcntx*necntx-1)
      logical*4 visusd,snglnt,uniform,ice
      byte eco_type,testbits(2)

      if(ice) then

        call PolarDay_snow(nbands,pxldat,vza,visusd,eco_type,avgtherm,
     *                     testbits,confdnc)

      else

        call ocean_day(nbands,pxldat,avgtherm,vza,pxrfa,snglnt,visusd,
     *                 eco_type,pw,testbits,confdnc)

c       If confidence is uncertain, apply the spatial variability test.
c       Also, make sure that the scene has uniform surface characteristics
c       (see reg_anc.f).

        if(uniform .and. confdnc.le.0.95 .and. confdnc.gt.0.34) then

c         Get brightness temperature differences between pixel
c         of interest and the ones surrounding it.
          call get_regdif(nlcntx,necntx,inband,nbands,rgdata,
     *                    diff)

c         Check variation in the region.
          call spatial_var(nbands,nlcntx,necntx,diff,varslt)

c         Bump up the confidence if spatial variability test showed
c         uniform
          if ((varslt .eq. 1) .and. (confdnc .gt. .66)) then
            confdnc = 0.96
          else if ((varslt .eq. 1) .and. (confdnc .le. 0.66)) then
            confdnc = 0.67
          endif

        end if

      end if

      return
      end
```

```
      subroutine water_nite(pxldat,vza,eco_type,ice,uniform,pw,
     *                      nlcntx,necntx,inband,nbands,
     *                      rgdata,avgtherm,diff,testbits,confdnc)


c     Routine for setting appropriate flags and processing path
c     for nighttime observations over water surfaces.

      integer*4 varslt
      real*4 pxldat(nbands),vza,confdnc,pw,avgtherm(nbands),
     *       rgdata(nlcntx,necntx,nbands),
     *       diff(nbands,nlcntx*necntx-1)
      logical*4 ice,uniform
      byte eco_type,testbits(2)

      if(ice) then

        call PolarNite_snow(nbands,pxldat,vza,eco_type,avgtherm,
     *                      testbits,confdnc)

      else

        call ocean_nite(nbands,pxldat,avgtherm,vza,pw,testbits,
     *                  confdnc)

c       If confidence is still uncertain, apply the spatial variability
c       test. Also check for regional scene uniformity (see reg_anc.f).

        if(uniform .and. confdnc.le.0.95 .and. confdnc.gt.0.34)then

c         Get brightness temperature differences between pixel
c         of interest and the ones surrounding it.
          call get_regdif(nlcntx,necntx,inband,nbands,rgdata,
     *                    diff)

c         Check variation in the region.
          call spatial_var(nbands,nlcntx,necntx,diff,varslt)

c         Bump up the confidence if spatial variability test showed
c         uniform
          if ((varslt .eq. 1) .and. (confdnc .gt. .66)) then
            confdnc = 0.96
          else if ((varslt .eq. 1) .and. (confdnc .le. 0.66)) then
            confdnc = 0.67
          endif

        end if

      end if

      return
      end
```

```
      subroutine write_bits(outf_unit,bitarray,iyrday,ibhms,islhms1,
     *                      outrec,end_file)

C!F77 ********************************************************
c ... routine for writing results of the cloud mask processing
c ... to a binary file.  the file is direct access with a 1432
c ... byte record length (16 bits or 2 1-byte words for each of 716
c ... MAS pixels in a scan line).  in the case of partial proccessing
c ... of scan lines, the file is padded with zeros
c ... The first line (record) contains nothing
c ... but 4 4-byte integers representing the beginning date (yyddd),
c ... time (hms), time in milliseconds and total number of records
c ... written to the file.
c ...
c!Input parameters:
c outf_unit             FORTRAN unit # attached to output file
c bitarray              array containing one scanline of bit results
c iyrday                Beginning date of data (yyddd)
c ibhms                 Beginning time of data (hhmmss)
c islhms1               Beginning time of cloud mask (hhmmss)
c outrec                Output record number
c end_fil               End of processing flag - set equal to
c                       .true. when finished
c!Output Parameters:
c ** none ** writes output to file
c
c!END*********************************************************

      include 'cldmask.inc'

c*************************************************************************
c ... scalar arguments ..
      integer*4 outf_unit
      integer*4 iyrday,ibhms,islhms1,outrec
      logical end_file
c ...
c ... array arguments ..
      byte bitarray(2,npixel)
c ...
c ... local arrays ..
      integer*4 hedr(npixel/2)

      save hedr

c ... On first call, write header record containing orbit start
c ... date and time, and cloud mask beginning time in milliseconds.

      if (outrec.eq.2) then

        hedr(1) = iyrday
        hedr(2) = ibhms
        hedr(3) = islhms1

        write (outf_unit,rec=1) hedr
        write (outf_unit,rec=outrec) bitarray

      else if (.not. end_file) then

        write (outf_unit,rec=outrec) bitarray

      end if

      if (end_file) then

c ... The end of data has been reached.  Write total number
c ... of records in the output file to the header.
        hedr(4) = outrec
        write (outf_unit,rec=1) hedr

      end if

      return
      end
```

```
      integer function write_proc(maxlin,outf_unit,ibls,nelin,bitarray,
     *                            iyrday,ibhms,
     *                            islhms1,outrec,procflg,block,count)

c*********************************************************************

      include 'cldmask.inc'

c*********************************************************************
c
c       external variables

      integer*4 maxlin  ! number of lines in the image data file
      integer*4 outf_unit       ! unit No. of cloud flag file
      integer*4 ibls            ! start line No.
      integer*4 nelin           ! end line No.
      byte bitarray(2,npixel,*) ! cloud flag data
      integer*4 iyrday          ! date of first scan line (ddd)
      integer*4 ibhms           ! time of first scan line (hhmmss)
      integer*4 islhms1         ! start time of the cloud mask (hhmmss)
      integer*4 outrec(outmax)  ! output record No.
      integer*4 procflg         ! procedure flag
      integer*4 block           ! block = (nelin-ibls)/linmax
      integer*4 count           ! count[0,block]

c*********************************************************************
c
c       internal variables

      logical*4 end_file
      data end_file /.false./
      integer*4 line
      integer*4 cnt,sline,eline
      integer*4 mlin /0/

c*********************************************************************

c           Write bit flags to output file for current scan line.

      if(procflg .eq. 0) then

c       write(*,'(1x,''#   file write '')')
C       write(6,*) 'sline=',sline,' eline=',eline

C        mlin = 0
        cnt = 0
        sline = ibls + linmax * count
        if(count.eq.block) then
           eline = nelin
        else
           eline = sline + linmax - 1
        endif

C        Do 100 line=ibls,nelin
        Do 100 line=sline,eline

C            write(6,*) 'line=',line,' mlin=',mlin
          if(line .le. maxlin) then

            mlin = mlin + 1
            cnt = cnt + 1
c            write(6,*) 'line=',line,' mlin=',mlin,' outrec=',outrec(line)

            if(mlin .eq. nlcntx) then

c             write(*,'(1x,''#   data write '')')
c             write(*,'(1x,''#   outrec= '',i10)') outrec(line)
c             write(*,'(1x,''#   islhms1= '',i10)') islhms1

              call write_bits(outf_unit,bitarray(1,1,cnt),iyrday,ibhms,
     *                        islhms1,outrec(line),end_file)

              mlin = nlcntx - 1

            endif

          end if

 100    continue

        endif

c*********************************************************************

        return
        end
```

```
/*  GIHLS211 transforms latitude, longitude coordinates to line, sample
    Renamed GIHLS211 to GETCOORD !!
    coordinates for an image in the Goode's Interrupted Homolosine
    projection.  This routine was complied and run using the C compiler on
    SunOS 4.2 (UNIX).  Results were accurate at the time of testing, but they
    are by no means guaranteed!

    By D. Steinwand, HSTX/EROS Data Center    June, 1993

References

1.  Snyder, John P. and Voxland, Philip M., "An Album of Map Projections",
    U.S. Geological Survey Professional Paper 1453 , United State Government
    Printing Office, Washington D.C., 1989.

2.  Snyder, John P., "Map Projections--A Working Manual", U.S. Geological
    Survey Professional Paper 1395 (Supersedes USGS Bulletin 1532), United
    State Government Printing Office, Washington D.C., 1987.

3.  Goode, J.P., 1925,  The Homolosine projection:  a new device for
    portraying the Earth's surface entire:  Assoc. Am. Geographers, Annals,
    v. 15, p. 119-125

4.  Steinwand, Daniel R., "Mapping Raster Imagery to the Interrupted
    Goode Homolosine Projection", 1993.  In press--IJRS.
-------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#define PI      3.141592653589793238
#define HALF_PI PI*0.5
#define TWO_PI  PI*2.0
#define EPSLN   1.0e-10
#define R2D     57.2957795131
#define D2R     0.0174532925199

#define OK      1
#define ERROR  -1
#define IN_BREAK -2

/* Variables common to all subroutines in this code file
  -----------------------------------------------------*/
static double R;                /* Radius of the earth (sphere) */
static double lon_center[12];   /* Central meridians, one for each region */
static double feast[12];        /* False easting, one for each region */

/* Transformation routine
  ---------------------------------*/
void getcoord_(double *lt, double *ln, double *lne, double *smp)
{
float pixsiz;
double x,y;
double lat, lon, line, samp;
int nl,ns;
double ul_x, ul_y;
int count;

pixsiz = 1000.0;

/* Report parameters and image geometric characteristics to the user
  -------------------------------------------------------------------*/
/*printf("Converting latitude, longitude to line, sample coordinates\n\n");
printf("Pixel size is %f km\n\n", pixsiz/1000.0);  */
if (pixsiz == 1000.0) {
   ul_x = -20015000.0;
   ul_y =   8673000.0;
   nl = 17347;
   ns = 40031;
   }
else {
   ul_x = -20011500.0;
   ul_y =   8669500.0;
   nl = 2168;
   ns = 5004;
   }
/*printf("Image size is %d lines by %d samples with an upper left\n",nl,ns);
printf("corner of UL_X = %lf and UL_Y = %lf meters.\n", ul_x, ul_y);   */

/* Initialize the Interrupted Goode Homolosine projection
  -----------------------------------------------------*/
goode_init(6370997.0);

/* Process point
  ------------------*/
   lat = *lt;
   lon = *ln;
/* printf("%12.6f %12.6f\n",lat, lon);   */
   lat *= D2R;
   lon *= D2R;
   goode_forward(lon, lat, &x, &y);
   line = (ul_y - y) / pixsiz + 1.0;
   samp = (x - ul_x) / pixsiz + 1.0;
/* printf("%12.6f %12.6f\n",lat, lon);
   printf("%12.6f %12.6f\n",line, samp);    */

   *lne = line;
   *smp = samp;
   *lt = lat;
   *ln = lon;

   return;
}

/* Function to report bad parameters
  ---------------------------------*/
bad_input_parms() {
printf("Syntax: gihl12ls pixsize\n");
printf("         pixsize in km = 1 or 8\n\n");
}

/* Initialize the Goode's Homolosine projection
  -------------------------------------------*/
goode_init(r)
double r;                       /* (I) Radius of the earth (sphere) */
{
/* Place parameters in static storage for common use
  -------------------------------------------------*/
R = r;

/* Initialize central meridians for each of the 12 regions
  -------------------------------------------------------*/
lon_center[0] = -1.74532925199;         /* -100.0 degrees */
lon_center[1] = -1.74532925199;         /* -100.0 degrees */
lon_center[2] =  0.523598775598;        /*   30.0 degrees */
lon_center[3] =  0.523598775598;        /*   30.0 degrees */
lon_center[4] = -2.79252680319;         /* -160.0 degrees */
lon_center[5] = -1.0471975512;          /*  -60.0 degrees */
lon_center[6] = -2.79252680319;         /* -160.0 degrees */
lon_center[7] = -1.0471975512;          /*  -60.0 degrees */
lon_center[8] =  0.349065850399;        /*   20.0 degrees */
lon_center[9] =  2.44346095279;         /*  140.0 degrees */
lon_center[10] = 0.349065850399;        /*   20.0 degrees */
lon_center[11] = 2.44346095279;         /*  140.0 degrees */

/* Initialize false eastings for each of the 12 regions
  ----------------------------------------------------*/
```

```
feast[0] = R * -1.74532925199;
feast[1] = R * -1.74532925199;
feast[2] = R * 0.523598775598;
feast[3] = R * 0.523598775598;
feast[4] = R * -2.79252680319;
feast[5] = R * -1.0471975512;
feast[6] = R * -2.79252680319;
feast[7] = R * -1.0471975512;
feast[8] = R * 0.349065850399;
feast[9] = R * 2.44346095279;
feast[10] = R * 0.349065850399;
feast[11] = R * 2.44346095279;

/* Report parameters to the user
  -----------------------------*/
/*ptitle("Goode's Homolosine Equal Area");
radius(r);    */
return(OK);
}
```

```
/* Goode's Homolosine forward equations--mapping lat,long to x,y
  ------------------------------------------------------------*/
goode_forward(lon, lat, x, y)
double lon;                     /* (I) Longitude */
double lat;                     /* (I) Latitude */
double *x;                      /* (O) X projection coordinate */
double *y;                      /* (O) Y projection coordinate */
{
double adjust_lon();    /* Function to adjust longitude to -180 - 180 */
double delta_lon;       /* Delta longitude (Given longitude - center */
double theta;
double delta_theta;
double constant;
int i;
int region;

/* Forward equations
  -----------------*/
if (lat >= 0.710987989993)                  /* if on or above 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 0;  /* If to the left of -40 */
   else region = 2;
   }
else if (lat >= 0.0)                        /* Between 0.0 and 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 1;  /* If to the left of -40 */
   else region = 3;
   }
else if (lat >= -0.710987989993)            /* Between 0.0 & -40 44' 11.8" */
   {
   if (lon <= -1.74532925199) region = 4;       /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 5; /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 8;    /* If between -20 and 80 */
   else region = 9;                             /* If between 80 and 180 */
   }
else                                            /* Below -40 44' */
   {
   if (lon <= -1.74532925199) region = 6;       /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 7;     /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 10;   /* If between -20 and 80 */
   else region = 11;                            /* If between 80 and 180 */
   }

if (region==1||region==3||region==4||region==5||region==8||region==9)
   {
   delta_lon = adjust_lon(lon - lon_center[region]);
   *x = feast[region] + R * delta_lon * cos(lat);
   *y = R * lat;
   }
else
   {
   delta_lon = adjust_lon(lon - lon_center[region]);
   theta = lat;
   constant = PI * sin(lat);

/* Iterate using the Newton-Raphson method to find theta
  -----------------------------------------------------*/
   for (i=0;;i++)
      {
      delta_theta = -(theta + sin(theta) - constant) / (1.0 + cos(theta));
      theta += delta_theta;
      if (fabs(delta_theta) < EPSLN) break;
      if (i >= 30)
         {giherror("Iteration failed to converge","Goode-forward");return(ERROR);}
      }
   theta /= 2.0;
   *x = feast[region] + 0.900316316158 * R * delta_lon * cos(theta);
```

```
   *y = R * (1.4142135623731 * sin(theta) - 0.0528035274542 * sign(lat));
   }

return(OK);
}

/* Functions to report projection parameters
  ----------------------------------------*/
ptitle(A) char *A; { printf("\n%s Projection Parameters:\n\n",A); }
radius(A) double A;{printf("   Radius of Sphere:      %lf meters\n",A); }

/* Function to report errors
  -------------------------*/
giherror(what, where) char *what,*where; {printf("[%s] %s\n",where,what);}

/* Function to calculate the sine and cosine in one call.  Some computer
   systems have implemented this function, resulting in a faster implementation
   than calling each function separately.  It is provided here for those
   computer systems which don't implement this function
  ----------------------------------------------------*/
#ifndef sun
sincos(val, sin_val, cos_val) double val; double *sin_val; double *cos_val;
{ *sin_val = sin(val); *cos_val = cos(val); return; }
#endif

/* Function to return the sign of an argument
  ------------------------------------------*/
sign(x) double x; { if (x < 0.0) return(-1); else return(1);}

/* Function to adjust longitude to -180 to 180
  ------------------------------------------*/
double adjust_lon(x) double x;{x=(fabs(x)<PI)?x:(x-(sign(x)*TWO_PI));return(x);}

/* Functions to compute constants e0,e1,e2, and M
  ------------------------------------------------*/
double e0fn(x) double x;{return(1.0-0.25*x*(1.0+x/16.0*(3.0+1.25*x)));}
double e1fn(x) double x;{return(0.375*x*(1.0+0.25*x*(1.0+0.46875*x)));}
double e2fn(x) double x;{return(0.05859375*x*x*(1.0+0.75*x));}
double mlfn(e0,e1,e2,phi) double e0,e1,e2,phi;{
                        return(e0*phi-e1*sin(2.0*phi)+e2*sin(4.0*phi));}
```

```
/*  GIHLS211 transforms latitude, longitude coordinates to line, sample
    Renamed GIHLS211 to GETCOORD !!
    coordinates for an image in the Goode's Interrupted Homolosine
    projection.  This routine was complied and run using the C compiler on
    SunOS 4.2 (UNIX).  Results were accurate at the time of testing, but they
    are by no means guaranteed!

    By D. Steinwand, HSTX/EROS Data Center    June, 1993

References

1.  Snyder, John P. and Voxland, Philip M., "An Album of Map Projections",
    U.S. Geological Survey Professional Paper 1453 , United State Government
    Printing Office, Washington D.C., 1989.

2.  Snyder, John P., "Map Projections--A Working Manual", U.S. Geological
    Survey Professional Paper 1395 (Supersedes USGS Bulletin 1532), United
    State Government Printing Office, Washington D.C., 1987.

3.  Goode, J.P., 1925,  The Homolosine projection:  a new device for
    portraying the Earth's surface entire:  Assoc. Am. Geographers, Annals,
    v. 15, p. 119-125

4.  Steinwand, Daniel R., "Mapping Raster Imagery to the Interrupted
    Goode Homolosine Projection", 1993.  In press--IJRS.
-------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>

#define PI      3.141592653589793238
#define HALF_PI PI*0.5
#define TWO_PI  PI*2.0
#define EPSLN   1.0e-10
#define R2D     57.2957795131
#define D2R     0.0174532925199

#define OK      1
#define ERROR  -1
#define IN_BREAK -2

/* Variables common to all subroutines in this code file
  -----------------------------------------------------*/
static double R;                /* Radius of the earth (sphere) */
static double lon_center[12];   /* Central meridians, one for each region */
static double feast[12];        /* False easting, one for each region */

/* Transformation routine
  -------------------------------*/
void nacoord_(double *lt, double *ln, double *lne, double *smp)
{
float pixsiz;
double x,y;
double lat, lon, line, samp;
int nl,ns;
double ul_x, ul_y;
int count;

pixsiz = 1000.0;

/* Report parameters and image geometric characteristics to the user
  -------------------------------------------------------------------*/
/*printf("Converting latitude, longitude to line, sample coordinates\n\n");
printf("Pixel size is %f km\n\n", pixsiz/1000.0);  */
if (pixsiz == 1000.0) {
   ul_x = -17359000.0;
   ul_y = 8423000.0;
   nl = 7793;
   ns = 11329;
   }
else {
   ul_x = -20011500.0;
   ul_y =   8669500.0;
   nl = 2168;
   ns = 5004;
   }
/*printf("Image size is %d lines by %d samples with an upper left\n",nl,ns);
printf("corner of UL_X = %lf and UL_Y = %lf meters.\n", ul_x, ul_y);   */

/* Initialize the Interrupted Goode Homolosine projection
  ------------------------------------------------------*/
ngoode_init(6370997.0);

/* Process point
  -----------------*/
   lat = *lt;
   lon = *ln;
/* printf("%12.6f %12.6f\n",lat, lon);    */
   lat *= D2R;
   lon *= D2R;
   ngoode_forward(lon, lat, &x, &y);
   line = (ul_y - y) / pixsiz + 1.0;
   samp = (x - ul_x) / pixsiz + 1.0;
/* printf("%12.6f %12.6f\n",lat, lon);
   printf("%12.6f %12.6f\n",line, samp);     */

   *lne = line;
   *smp = samp;
   *lt = lat;
   *ln = lon;

   return;
}

/* Function to report bad parameters
  ---------------------------------*/
nbad_input_parms() {
printf("Syntax: gihl12ls pixsize\n");
printf("         pixsize in km = 1 or 8\n\n");
}

/* Initialize the Goode's Homolosine projection
  --------------------------------------------*/
ngoode_init(r)
double r;                       /* (I) Radius of the earth (sphere) */
{
/* Place parameters in static storage for common use
  -------------------------------------------------*/
R = r;

/* Initialize central meridians for each of the 12 regions
  -------------------------------------------------------*/
lon_center[0] = -1.74532925199;         /* -100.0 degrees */
lon_center[1] = -1.74532925199;         /* -100.0 degrees */
lon_center[2] =  0.523598775598;        /*   30.0 degrees */
lon_center[3] =  0.523598775598;        /*   30.0 degrees */
lon_center[4] = -2.79252680319;         /* -160.0 degrees */
lon_center[5] = -1.0471975512;          /*  -60.0 degrees */
lon_center[6] = -2.79252680319;         /* -160.0 degrees */
lon_center[7] = -1.0471975512;          /*  -60.0 degrees */
lon_center[8] =  0.349065850399;        /*   20.0 degrees */
lon_center[9] =  2.44346095279;         /*  140.0 degrees */
lon_center[10] = 0.349065850399;        /*   20.0 degrees */
lon_center[11] = 2.44346095279;         /*  140.0 degrees */

/* Initialize false eastings for each of the 12 regions
  ----------------------------------------------------*/
```

```
feast[0] = R * -1.74532925199;
feast[1] = R * -1.74532925199;
feast[2] = R * 0.523598775598;
feast[3] = R * 0.523598775598;
feast[4] = R * -2.79252680319;
feast[5] = R * -1.0471975512;
feast[6] = R * -2.79252680319;
feast[7] = R * -1.0471975512;
feast[8] = R * 0.349065850399;
feast[9] = R * 2.44346095279;
feast[10] = R * 0.349065850399;
feast[11] = R * 2.44346095279;

/* Report parameters to the user
  -----------------------------*/
/*nptitle("Goode's Homolosine Equal Area");
nradius(r);   */
return(OK);
}
```

```
/* Goode's Homolosine forward equations--mapping lat,long to x,y
  ----------------------------------------------------------------*/
ngoode_forward(lon, lat, x, y)
double lon;                     /* (I) Longitude */
double lat;                     /* (I) Latitude */
double *x;                      /* (O) X projection coordinate */
double *y;                      /* (O) Y projection coordinate */
{
double nadjust_lon();   /* Function to adjust longitude to -180 - 180 */
double delta_lon;       /* Delta longitude (Given longitude - center */
double theta;
double delta_theta;
double constant;
int i;
int region;

/* Forward equations
  -----------------*/
if (lat >= 0.710987989993)                  /* if on or above 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 0;  /* If to the left of -40 */
   else region = 2;
   }
else if (lat >= 0.0)                        /* Between 0.0 and 40 44' 11.8" */
   {
   if (lon <= -0.698131700798) region = 1;  /* If to the left of -40 */
   else region = 3;
   }
else if (lat >= -0.710987989993)            /* Between 0.0 & -40 44' 11.8" */
   {
   if (lon <= -1.74532925199) region = 4;       /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 5; /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 8;    /* If between -20 and 80 */
   else region = 9;                             /* If between 80 and 180 */
   }
else                                            /* Below -40 44' */
   {
   if (lon <= -1.74532925199) region = 6;       /* If between -180 and -100 */
   else if (lon <= -0.349065850399) region = 7;     /* If between -100 and -20 */
   else if (lon <= 1.3962634016) region = 10;   /* If between -20 and 80 */
   else region = 11;                            /* If between 80 and 180 */
   }

if (region==1||region==3||region==4||region==5||region==8||region==9)
   {
   delta_lon = nadjust_lon(lon - lon_center[region]);
   *x = feast[region] + R * delta_lon * cos(lat);
   *y = R * lat;
   }
else
   {
   delta_lon = nadjust_lon(lon - lon_center[region]);
   theta = lat;
   constant = PI * sin(lat);

/* Iterate using the Newton-Raphson method to find theta
  -----------------------------------------------------*/
   for (i=0;;i++)
      {
      delta_theta = -(theta + sin(theta) - constant) / (1.0 + cos(theta));
      theta += delta_theta;
      if (fabs(delta_theta) < EPSLN) break;
      if (i >= 30)
         {ngiherror("Iteration failed to converge","Goode-forward");return(ERROR);}
      }
   theta /= 2.0;
   *x = feast[region] + 0.900316316158 * R * delta_lon * cos(theta);
```

```
    *y = R * (1.414213562731 * sin(theta) - 0.0528035274542 * nsign(lat));
    }

return(OK);
}

/* Functions to report projection parameters
  ----------------------------------------*/
nptitle(A) char *A; { printf("\n%s Projection Parameters:\n\n",A); }
nradius(A) double A;{printf("   Radius of Sphere:      %lf meters\n",A); }

/* Function to report errors
  -------------------------*/
ngiherror(what, where) char *what,*where; {printf("[%s] %s\n",where,what);}

/* Function to calculate the sine and cosine in one call.  Some computer
   systems have implemented this function, resulting in a faster implementation
   than calling each function separately.  It is provided here for those
   computer systems which don't implement this function
  ---------------------------------------------------*/
#ifndef sun
nsincos(val, sin_val, cos_val) dpuble val; double *sin_val; double *cos_val;
{ *sin_val = sin(val); *cos_val = cos(val); return; }
#endif

/* Function to return the sign of an argument
  ------------------------------------------*/
nsign(x) double x; { if (x < 0.0) return(-1); else return(1);}

/* Function to adjust longitude to -180 to 180
  -------------------------------------------*/
double nadjust_lon(x) double x;{x=(fabs(x)<PI)?x:(x-(nsign(x)*TWO_PI));return(x);}

/* Functions to compute constants e0,e1,e2, and M
  ----------------------------------------------*/
double ne0fn(x) double x;{return(1.0-0.25*x*(1.0+x/16.0*(3.0+1.25*x)));}
double ne1fn(x) double x;{return(0.375*x*(1.0+0.25*x*(1.0+0.46875*x)));}
double ne2fn(x) double x;{return(0.05859375*x*x*(1.0+0.75*x));}
double nm1fn(e0,e1,e2,phi) double e0,e1,e2,phi;{
                        return(e0*phi-e1*sin(2.0*phi)+e2*sin(4.0*phi));}
```